



Thèse

présentée pour obtenir le grade de docteur
de l'Ecole Nationale Supérieure des Télécommunications

Spécialité **: Informatique et Réseaux**

Jacques DEMERJIAN

Services d'autorisation et Intégration au protocole d'attribution dynamique des adresses

Soutenue le 09/12/2004 devant le jury composé de :

Pierre PARADINAS	Président
Abdelmadjid BOUABDALLAH	Rapporteurs
Éric HORLAIT	
Dominique SERET	Examinateurs
Guy PUJOLLE	
Mohamed ACHEMLAL	
Sylvain GOMBAULT	
Ahmed SERHROUCHNI	Directeur de thèse

à mes parents.

Remerciements

Je tiens à remercier, et en premier lieu, **Dieu** pour toutes ces bénédictions qu'il m'a offertes.

Je remercie mes parents « Abou Jacques : Baron Georges » et « Em Jacques : Albera » qui, grâce à leurs sacrifices et dévouement, j'ai réussi à décrocher ce titre de doctorat.

Je remercie mes sœurs « Olivia et Nirma » pour leur soutien durant toutes mes années d'études. Je suis reconnaissant envers mon beau-frère « Rafi » pour son accueil et pour toutes ses aides durant mon séjour en France. Je remercie mon beau-frère « Georges » pour tout ce qu'il a fait pour moi.

Je remercie « Karine » pour sa présence permanente et son soutien durant les périodes difficiles de ma thèse.

Je tiens à remercier, Monsieur Ahmed Serhrouchni qui a accepté de me suivre pendant mes années de recherche. Je le remercie pour sa fraternité, pour nos discussions fructueuses, ses conseils pertinents et sa longue patience.

Je suis reconnaissant envers Monsieur Eric Horlait et Monsieur Abdelmadjid Bouabdallah qui ont bien donné de leur temps afin d'être rapporteurs de cette thèse, et aussi envers Monsieur Mohammed Achemlal, Monsieur Sylvain Gombault, Monsieur Pierre Paradinas, Monsieur Guy Pujolle et Madame Dominique Serret qui ont porté la lourde tâche d'être membres du jury.

J'exprime ma gratitude à Ibrahim Hajjeh pour tout le support qu'il m'a offert pendant les années de travail, à Ouahiba Fouial et Salim Ferraz pour toutes leur aides durant ma thèse, surtout d'avoir été présents durant les moments difficiles.

Je remercie mes ami(e)s : Mohammad, Toto, Pierre, Dani, Céline, Rani, Ahmad, Nadine, Rana, Magdi, Katia, Rima, Michel, Kinda, Hazar, $Rami^2$, Xue, Rola, Fadi, Achraf,... pour leur soutien moral.

Enfin, ils ont été toujours à mes côtés, ils sont tout pour moi, c'est à eux que je dédie ce rapport de thèse. A mon père, à ma mère, à mes sœurs et surtout à la mémoire de ma grand-mère « Nadira » qui a souhaité me voir docteur mais qui n'a pas réussi à nous partager cette joie.

Jacques DEMERJIAN

Résumé

La sécurité est un enjeu majeur des technologies numériques modernes. Avec le développement de l'Internet, les besoins de sécurité sont de plus en plus importants. Le développement d'applications Internet telles que le commerce électronique, les applications médicales ou la vidéoconférence, implique de nouveaux besoins comme, l'identification des entités communicantes, l'intégrité des messages échangés, la confidentialité de la transaction, l'authentification des entités, l'anonymat du propriétaire du certificat, l'habilitation des droits, la procuration, etc..

Qu'il s'agisse de données médicales, fiscales ou bancaires, le besoin en sécurité est essentiel afin de crédibiliser le système, tout en respectant à la fois les besoins des utilisateurs et des applications. Cette sécurité a néanmoins un prix : celui de l'établissement de la confiance entre les partenaires en communication. La confiance des utilisateurs passe par la sécurisation des transactions, par exemple au moyen d'une procédure de certification, et la reconnaissance des signatures électroniques.

Malgré la diversité des certificats numériques existants (certificat d'identité X.509, *SPKI*, certificat d'attributs, etc.), ils sont encore limités, génériques et répondent ainsi insuffisamment aux besoins spécifiques des applications électroniques et des utilisateurs. D'où la nécessité de spécifier une nouvelle approche pour la génération de certificats numériques répondant à ces exigences, légers, simplifiés et plus ouverts que ceux existants.

Les travaux de recherche présentés dans cette thèse consistent à proposer une nouvelle approche pour la génération de certificats numériques pour contribuer aux services d'autorisation, puis à intégrer cette contribution au protocole d'attribution dynamique des adresses *DHCP* (*Dynamic Host Configuration Protocol*) afin de le renforcer.

Cette thèse est constituée de deux parties.

Dans la première partie, nous traitons les différents types de certificats existants ainsi que leurs limites. Nous proposons et spécifions une approche qui permet de garantir à l'application et à l'utilisateur la bonne mise en forme des informations dans le certificat et l'adéquation du contenu de leur certificat vis-à-vis de leurs besoins. Ces certificats sont des certificats d'attributs spécifiés en *XML*, flexibles et respectant les besoins de l'application et la personnalisation de l'utilisateur durant la génération du certificat.

Pour chaque application, nous avons défini une grammaire *DTD* (*Document Type Definition*) pour préciser tous les champs dont l'application a besoin. L'idée principale est de stocker, sur le serveur, des *DTDs* c'est-à-dire, des fichiers contenant un certain nombre de paramètres correspondant aux données qui seront insérées dans le certificat d'attributs final. La génération de ces certificats d'attributs respecte la grammaire associée à l'application. En effet, c'est grâce à celles-ci que l'administrateur personnalisera les certificats d'attributs que l'utilisateur pourra demander. Ainsi, si le besoin d'un nouveau type de certificat d'attributs émane, il suffit de créer la *DTD* correspondant à la nouvelle application ajoutée.

Pour satisfaire les besoins de l'utilisateur, l'*E-IGP* (Extension de l'Infrastructure de Gestion des Privilèges) permet à ce dernier de personnaliser sa demande de certificats d'attributs. C'est l'utilisateur qui précise les valeurs des paramètres de l'application, la date de validité de son certificat d'attributs, les rôles qu'il souhaite avoir ou les délégations qu'il souhaite fournir à quelqu'un suivant ces besoins. La mise en œuvre de l'*E-IGP* a nécessité l'existence d'une Infrastructure de Gestion des Clefs, à laquelle l'*E-IGP* est rattaché.

Pour prouver la faisabilité et l'efficacité de l'approche proposée, nous l'intégrons dans le fonctionnement du protocole *DHCP*. Destiné à faciliter le travail des administrateurs systèmes en automatisant l'attribution des adresses IP et les paramètres de configurations aux clients du réseau, le protocole *DHCP* souffre de nombreux problèmes de sécurité. Il ne supporte pas le mécanisme avec lequel les clients et les serveurs *DHCP* s'authentifient. De plus, le protocole *DHCP* n'assure pas l'intégrité des données échangées, ni leur confidentialité et il ne possède aucun mécanisme de contrôle d'accès.

La deuxième contribution majeure de cette thèse est la spécification et l'implémentation d'une extension du protocole *DHCP*, appelée *E-DHCP* (*Extended Dynamic Host Configuration Protocol*). *E-DHCP* présente une méthode d'authentification d'entités (client et serveur) *DHCP* et des contenus des messages *DHCP*. *E-DHCP* propose une nouvelle option *DHCP*. La technique utilisée par cette option est basée sur l'utilisation d'algorithmes de clefs de chiffrement asymétrique, de certificats d'identité X.509 et de certificats d'attributs simplifiés spécifiés en *XML*, proposés dans la première contribution de cette thèse. L'idée principale de *E-DHCP* est d'adosser au serveur *DHCP* un serveur *AA* (*Attribute Authority*) d'un *E-IGP* pour former un nouveau serveur appelé serveur *E-DHCP*. Ce nouveau serveur crée un certificat d'attributs pour le client contenant l'adresse Internet attribuée dynamiquement. L'utilisation du certificat d'attributs confirme la possession du client de son adresse IP.

Abstract

Security is a major stake of modern numerical technologies. With the development of the Internet, the needs for security are increasingly becoming more important than ever before. The development of Internet applications such as e-business, medical applications or videoconference, implies new needs such as, identification of the communicating entities, integrity of the exchanged messages, confidentiality of the transaction, authentication of the entities, anonymity of the certificate owner, rights capacitation, procuration, etc.

Whether they are medical, tax or banking related data, the requirement in security is essential in order to give credibility to the system, while respecting at the same time users' and applications' needs. This security has nevertheless a price: that of the establishment of trust between communicating partners. Users' trust goes through transactions security, for example by means of a certification procedure, and the recognition of electronic signatures.

Despite the diversity of existing certificates (X.509 identity certificate, *SPKI*, attributes certificate, etc), they are still limited, generic and thus, meet insufficiently the specific needs of electronic applications and users. Hence, the need for specifying a new approach for the generation of certificates answering these requirements, light, simplified and more open than those existing.

The research tasks presented in this thesis consist in proposing a new approach for the generation of certificates to contribute to the authorization services, then to integrate this contribution to *DHCP* (Dynamic Host Configuration Protocol) in order to reinforce it.

This thesis consists of two parts.

In the first part, we treat various types of existing certificates as well as their limits. We propose and specify an approach which makes it possible to guarantee to the application and to the users a good representation of information within the certificate and the adequacy of the certificate content with their needs. These certificates are attributes certificates specified in XML, flexible and respecting the needs of the application and user's personalization during the certificate generation process.

For each application, we defined a grammar *DTD* (*Document Type Definition*) to specify all required fields for the application. The main idea is to store, on the server, *DTDs* i.e. files containing a certain number of parameters corresponding to data that will be inserted in the final attributes certificate. The generation of these attributes certificates respects the application associated grammar. Indeed, it is thanks to these parameters that the administrator will personalize the attributes certificates that the user will be able to request. Thus, if the need for a new type of attributes certificate emanates, it would be enough to create the *DTD* corresponding to the new added application.

To satisfy users' needs, *E-IGP* (in English *E-PMI* for Extended-Privilege Management Infrastructure) allows the use to customize her/his attributes certificate request, the user specifies the application parameters' values, the validity period of the attributes certificate, roles that she/he would like to have or the delegation she/he would like to provide someone else according to these needs. The set of *E-IGP* (*E-PMI*) required the existence of a Key Management Infrastructure, to which the *E-IGP* (*E-PMI*) is bound.

To prove the feasibility and the effectiveness of the suggested approach, we integrate it in the functioning of *DHCP*. Intended to facilitate the work of the system administrators by automating the attribution of IP addresses and the configuration parameters to the network clients, *DHCP* suffers from many security problems. It does not support the mechanism with which *DHCP* client and server are

authenticated. Moreover, *DHCP* does not ensure the integrity of the exchanged messages, nor their confidentiality, and it does not have any access control mechanism.

The second major contribution of this thesis is the specification and the implementation of an extension to *DHCP*, called *E-DHCP* (*Extended Dynamic Host Configuration Protocol*). *E-DHCP* presents a method of authentication of *DHCP* entities (client and server) and of *DHCP* messages contents. *E-DHCP* proposes a new DHCP option. The technique used by this option is based on the use of asymmetric key algorithms, of X.509 identity certificates and of simplified attributes certificates specified in *XML*, suggested in the first contribution of this thesis. The main idea of E-*DHCP* is to support *DHCP* server with an *AA* (Attribute Authority) server of an *E-PMI* to form a new server called *E-DHCP* server. This new server creates an attributes certificate for the client containing the Internet address dynamically allocated. The use of the attributes certificate confirms the possession of the client of their IP address.

Table des matières

Liste des figures

Liste des tableaux

PARTIE I

Services d’autorisation et
Intégration au protocole d’attribution dynamique des adresses

Chapitre 1

1 Introduction générale

1.1 Contexte de la thèse

De nouveaux besoins tels que, identification des entités communicantes, intégrité des messages échangés, confidentialité de la transaction, authentification des entités, etc., liés à la sécurité des communications sont apparus avec le développement de l'Internet et en particulier, celui du commerce électronique. En effet, son développement dépend de l'établissement de relations de confiance entre les parties de l'échange. Le manque de sécurité est souvent indiqué comme le principal obstacle au développement du commerce électronique. De ce fait, les données circulant sur le réseau Internet peuvent être interceptées. La confiance des utilisateurs passe donc par la sécurisation des transactions, par exemple au moyen d'une procédure de certification, et la reconnaissance des signatures électroniques.

Les certificats numériques [RFC1114] permettent d'établir un environnement de confiance entre deux entités distantes (deux personnes physiques, une personne et un serveur web, etc.) qui ont besoin de communiquer entre elles et de s'échanger des informations non répudiables par le biais d'application des signatures électroniques et des informations confidentielles par le biais de l'application de chiffrement des informations échangées. Le certificat numérique est indissociable des fonctions de signature électronique (identification de l'émetteur, intégrité du message, garantissant ainsi la non-répudiation) et de chiffrement (confidentialité d'un message). Le certificat numérique peut également servir pour l'authentification lors de besoins de contrôle d'accès.

Il existe différents types de certificats numériques (Certificat d'identité X.509 [RF3280], *SPKI* [RFC2692] [RFC2693], Certificat d'Attributs [RFC3281], *PGP* [RFC1991].). Malgré la souplesse ou les avantages de standardisation de ces certificats, aucun parmi eux ne répondent à tous les besoins apparus avec le développement de l'Internet.

Les certificats d'identité X.509 sont crées et gérées par l'IGC (Infrastructures de Gestion des Clefs) [RFC3280]. L'IGC tente de répondre à quatre besoins de sécurité des échanges. Premièrement, l'identification et l'authentification de deux parties en présence d'un tiers de confiance garantissant que son cyber-interlocuteur est bien celui qu'il prétend être. Deuxièmement, l'intégrité des messages transmis, un sceau permettant d'être sûr que des documents ne sont pas modifiés lorsqu'ils traversent un réseau. Troisièmement, la confidentialité de la transaction, la sécurisation pouvant porter sur le canal de transmission ou sur le message lui-même. Enfin, la non-répudiation de l'échange, c'est-à-dire l'impossibilité pour une partie de nier avoir signé ou reçu un document.

Les certificats d'identité X.509 sont orientés vers l'autentification. Le format des certificats le plus utilisé aujourd'hui dans le cadre d'une IGC est le format normalisé par le standard X.509 dans sa version trois [RFC3280]. X.509v3 est plus flexible que ces précédentes versions grâce au champs *extensions* qui permettent d'imposer plus de restrictions. Les extensions de certificats X.509v3 sont des champs qui permettent de rendre l'utilisation des certificats plus flexible. Néanmoins, elles rendent bien souvent les applications traitant ces certificats incompatibles entre elles. La prolifération de ces extensions génère des problèmes dans les domaines d'interopérabilité et de révocation. L'interopérabilité vienne du fait qu'une extension dans un certificat peut être qualifiée de critique ou

non. Le fait qu'une extension soit critique rend obligatoire la conformité du certificat aux informations contenues dans l’extension. Si une application traitant un certificat ne reconnaît pas une extension contenue dans ce certificat, et elle est marquée comme critique, ce certificat doit être déclaré invalide par l'application.

L’agrégation des attributs a comme conséquence le raccourcissement de la durée de vie du certificat, puisque chaque attribut a sa propre durée de vie. Ainsi, du fait que les attributs possèdent des durées de vie courtes et que leur contenu change plus souvent que l’identité, la fréquence de révocation du certificat augmente.

La structure des certificats d’identité X.509v3 telle qu’elle est définie dans le standard X.509 est commune à tous les exigences et couvre trop de besoins. Cette structure est "lourde" à mettre en œuvre, voire inadaptée pour certaines applications, notamment au niveau du codage.

Pour le commerce électronique comme pour l’éducation, il y a des besoins diversifiés (autres que ceux déjà vus: l’authentification, l'identification, etc.), par exemple des besoins de procuration, d’autorisation, d’anonymat du propriétaire du certificat, etc.. D’où la nécessité d’avoir des certificats répondant à ces exigences, "léger" et simplifiés.

Pour dépasser les limites que les certificats X.509v3 possèdent, un premier effort avait débuté à l’IETF (*International Enginering Task Force*) [IETF] pour définir des "certificats simplifiés" *SPKI* (*Simple Public Key Infrastructure*) [RFC2692]. Cependant, à trop vouloir simplifier, cette démarche ne convient plus à tous les besoins réels des utilisateurs et des applications à savoir l’authentification, l’identification, l’attribution de rôle, la procuration, l’habilitation, etc.. Ces certificats *SPKI* sont orientés seulement vers l’autorisation et l’anonymat du propriétaire du certificat.

Aujourd'hui, le concept de certification évolue naturellement à la gestion des attributs, des droits ou des privilèges d'un utilisateur. Ces droits sont, par exemple, l'autorisation d'accéder à des informations, l'appartenance à un groupe, un rôle, etc. La dernière version de X.509 [RFC3281], éditée en 2002, est la première édition pour normaliser une technique d'autorisation basée sur des certificats d'attributs et des IGP (Infrastructures de Gestion de Privilège). L’utilisation du certificat d’attributs rend possible la délégation des droits, la signature avec un rôle et le contrôle de la multisignature électronique. Bien que les certificats d’attributs aient résolu et simplifié les problèmes de certificats d’identité X.509v3, ils possèdent toujours des points faibles. Une part de ces faiblesses est attribuable à l’encodage des certificats d’attributs en format *ASN.1* [ITUT-X208-88] et à la difficulté d’intégration de nouveaux attributs au certificat. Jusqu’à ce jour, les champs du certificat d'attributs spécifiés dans le standard sont fixés et ne correspondent pas toujours aux besoins de l'utilisateur et/ou de l’application. C’est pourquoi, il s’est avéré nécessaire de trouver une solution qui puisse répondre à la fois aux vrais besoins de l’application (en précisant les paramètres dont l’application a besoin) et de l’utilisateur (en précisant la valeur des paramètres de l’application, les délégations, les rôles, la validité du certificat, etc).

Devant la diversité des besoins des domaines d’application professionnelle, éducative et privée, apparaît la nécessité de spécifier une nouvelle approche pour la génération de certificats numériques simplifiés plus ouverts que ceux existants. Dans la première partie de notre rapport, nous spécifions notre approche qui permet de garantir à l’application et à l’utilisateur la bonne mise en forme des informations dans le certificat, l’adéquation du contenu de leur certificat vis-à-vis de leur besoin. Ces certificats sont des certificats d’attributs spécifiés en *XML*, flexibles et respectant : la grammaire associée à chaque application et la personnalisation de l’utilisateur durant la génération du certificat. La génération de ces certificats d’attributs nécessitent l’existence d’une infrastructure pour gérer ces certificats. Ainsi, nous proposons et implémentons une infrastructure de gestion de ces certificats, appelée (E-IGP). E-IGP est une extension de l’Infrastructure de Gestion de Privilège.

Pour prouver la faisabilité et l’efficacité de l’approche proposée, nous intégrons notre approche dans le fonctionnement du protocole *DHCP* (*Dynamic Host Configuration Protocol*) [RFC2131].

Le protocole d'attribution dynamique des adresses Internet *DHCP* est nécessaire pour le fonctionnement d'un grand nombre de réseaux. Cela est dû notamment au manque d'adresses Internet qui ne permet pas leur attribution statique et au fait que la mobilité des équipements est plus adaptée à l'adresse dynamique. Destiné à faciliter le travail des administrateurs systèmes en automatisant l'attribution des adresses IP et les paramètres de configurations aux clients du réseau, le protocole *DHCP* souffre de nombreux problèmes de sécurité. Le protocole *DHCP* ne supporte pas le mécanisme avec lequel les clients et les serveurs *DHCP* s'authentifient. De plus, le protocole *DHCP* n'assure pas l'intégrité des données échangées, ni leur confidentialité. Le protocole *DHCP* n'a aucun mécanisme de contrôle d'accès.

Quelques solutions ont tenté de renforcer la sécurité du protocole *DHCP* afin d'assurer l'authentification d'entité (client et/ou serveur) *DHCP* et/ou des contenus des messages *DHCP*. Certaines de ces solutions utilisent la cryptographie, d'autres ne l'utilisent pas. Ces solutions possèdent des limites et sont, de plus, vulnérables à certains types d'attaques.

La deuxième contribution majeure de cette thèse est donc la spécification et l'implémentation d'une extension du protocole *DHCP*, appelée *E-DHCP* (*Extended Dynamic Host Configuration Protocol*).

E-DHCP présente une méthode d'authentification d'entité (client et serveur) *DHCP* et des contenus des messages *DHCP*. *E-DHCP* propose une nouvelle option *DHCP*. La technique utilisée par cette option est basée sur l'utilisation d'algorithme de clefs de chiffrement asymétrique, de certificats d'identité X.509 et de certificats d'attributs simplifiés spécifiés en *XML*, proposés dans la première contribution de cette thèse.

L'idée principale de *E-DHCP* est d'adosser au serveur *DHCP* un serveur *AA* (*Attribute Authority*) d'un E-IGP pour former un nouveau serveur appelé, serveur *E-DHCP*. Le serveur *E-DHCP* crée un certificat d'attributs simplifiés spécifiés en *XML* pour le client contenant l'adresse Internet attribuée dynamiquement. Ce certificat d'attributs sera ensuite utilisé dans le contrôle d'accès.

1.2 Cadre général et objectifs de la thèse

Les travaux de recherche présentés dans cette thèse consistent à proposer de nouvelles fonctionnalités et de nouveaux mécanismes pour contribuer à l'authentification forte et aux services d'autorisation, puis à intégrer cette contribution au protocole *DHCP* (*Dynamic Host Configuration Protocol*) [RFC2131] afin de le renforcer. Ces travaux ont été éffectués dans le cadre du projet RNRT ICARE [ICARE].

Les travaux réalisés dans cette thèse sont centrés sur les deux axes suivants (voir figure 1-1).

Le premier axe consiste à contribuer à l'authentification forte et aux services d'autorisation. Cet axe concerne la proposition et l'implémentation d'une extension de l'infrastructure de gestion des privilèges et d'autorisations basée sur des certificats d'attributs simplifiés. Le modèle proposé est appelé *E-IGP* (*Extension de l'Infrastructure de Gestion des Privilèges*) ou *E-PMI* (*Extended Privilege Managemant Infrastructure*) en anglais. *E-IGP* permet aux propriétaires et aux concepteurs de portails de marchés électroniques et d'entreprises ; d'utiliser une infrastructure de gestion des privilèges sécurisés et fournit aux clients des services d'accès sûrs et personnalisés. Les certificats d'attributs simplifiés créés par l'*E-IGP* satisfont à la fois les besoins spécifiques de l'application et de l'utilisateur.

Le deuxième axe consiste à consolider la sécurité du protocole d'attribution dynamique *DHCPv4 (Dynamic Host Configuration Protocol)* [RFC2131] en se basant sur la proposition présenté dans le premier axe de recherche. Cet axe concerne la proposition, la conception et l'implémentation d'une extension du protocole *DHCPv4*. Cette extension, appelée *E-DHCP* (*Extended-Dynamic Host Configuration Protocol*), permet d'assurer un contrôle strict sur les équipements par une procédure

d'authentification forte. Cette extension consiste à rajouter une nouvelle option *DHCP* qui fournit simultanément l'authentification d'entités et de messages *DHCP* en se basant sur l'utilisation de clefs de chiffrement asymétrique, de certificats d'identité X.509 et de certificats d'attributs. L'implémentation de *E-DHCP* est basée sur l'utilisation des fonctionnalités du serveur d'autorité d'attribution (*Atributes Authoritie*, *AA*) de l'E-PMI proposée dans le premier axe de recherche.

Les deux contributions *E-PMI* et *E-DHCP* sont basées sur une *API* cryptographique (voir figure 1-1) qui assure toutes les fonctions cryptographiques dont nous avons besoin pour le bon fonctionnement de ces deux propositions. *E-DHCP* utilise les certificats d'attributs qui sont spécifiés dans le *E-PMI*.

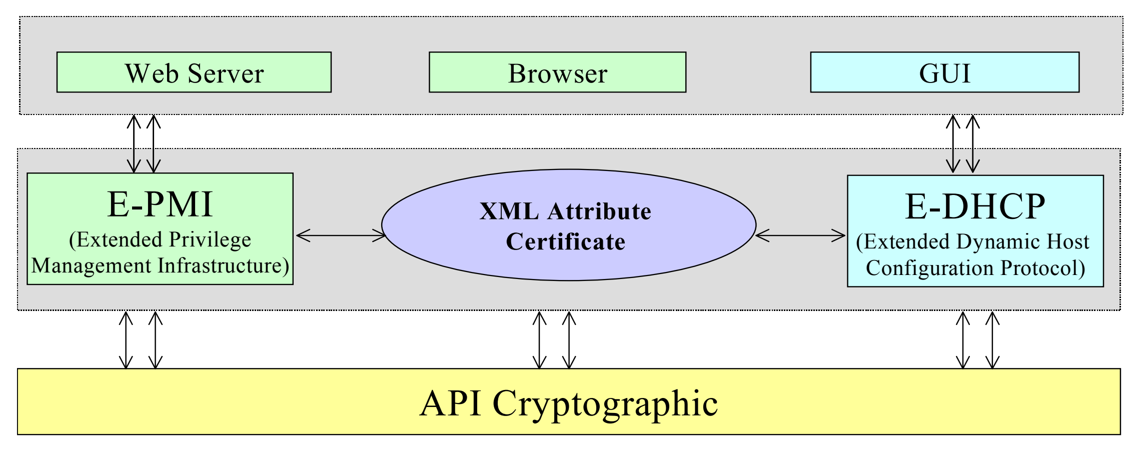


Figure 1-1 Intégration des travaux de recherches proposés dans cette thèse

1.3 Plan du rapport

Ce rapport de thèse est divisé en deux parties. La première partie présente notre contribution à l'authentification forte et aux services d'autorisation. La deuxième partie présente la conception d'une extension du protocole *DHCP* basée sur les certificats d'attributs. Cette conception est réalisée à travers l'intégration de notre première contribution au protocole *DHCP*.

La première partie est elle-même divisée en quatre chapitres. Dans le deuxième chapitre, nous introduisons les certificats d'identité X.509 et les Infrastructures de Gestions des Clefs (*IGC*) en focalisant la présentation sur les certificats d'identité X.509v3 ainsi que sur leurs limites. Cette étude est nécessaire pour montrer la lourdeur de la structure des certificats d'identité X.509 et de sa mise en œuvre.

Dans le troisième chapitre, nous introduisons les Infrastructures de Gestions de Privilèges (*IGP*) en se focalisant sur les certificats d'attributs, leur usage, leur intérêt et leurs limites.

Dans le quatrième chapitre, nous proposons une extension de l'Infrastructure de Gestion des Privilèges et d'autorisations basée sur des certificats d'attributs simplifiés appellée *E-IGP* (*Extension de l'Infrastructure de Gestion des Privilèges*). Ces certificats d'attributs satisfont à la fois les besoins spécifiques de l'application et de l'utilisateur. De même, nous présentons l'implémentation de cette extension.

La deuxième partie de notre rapport est composée de quatre chapitres. Dans le cinquième chapitre, nous décrivons le protocole *DHCP,* son fonctionnement, les failles et les problèmes de sécurité de *DHCP*. Nous illustrons ensuite les solutions existantes pour renforcer la sécurité de *DHCP* ainsi que leurs limites. Le sixième chapitre expose notre proposition appelée *E-DHCP*, son principe, ses conditions, ses scénarios et ses avantages. Dans le septième chapitre, nous modélisons *E-DHCP* en *UML* et nous présentons l'implémentation et les tests du protocole *E-DHCP*.

Nous présentons la synthèse de ce rapport dans le huitième chapitre avec l'ensemble des résultats et perspectives de nos travaux de recherche.

Chapitre 2

2 Les certificats d’identité X.509 et leurs limites

Résumé

Le chiffrement, et en particulier le chiffrement asymétrique, représente l’une des pistes les plus prometteuses pour avancer dans la sécurisation et l’authentification des échanges. L’Infrastructure de Gestions des Clefs (IGC) est une réponse conçue pour assurer la sécurité des transactions électroniques et permettre l’échange de renseignements sensibles entre des parties qui n’ont pas établi au préalable de liens entre elles. L’IGC est principalement destinée à émettre et gérer les certificats de clef publique (certificats d’identité). Le format normalisé admis sur le plan international pour les certificats de clef publique est défini dans la norme X.509. Actuellement, la plupart des certificats utilisés en pratique sont des certificats X.509v3. En effet, un certificat X.509v3 est une structure de données pouvant comporter de nombreuses extensions et de nombreuses options. Cette particularité permet l’utilisation des certificats X.509 pour beaucoup de situations et d’environnements mais rend difficile la création de produits interopérables. Ce chapitre présente, d’une part, une vision globale des IGCs et de ses composants et d’autre part, il introduit les certificats d’identité X.509 en focalisant la présentation sur les certificats d’identité X.509 version 3 ainsi que leurs limites.

2.1 Introduction

L’adoption de solutions basées sur la cryptographie à clefs publiques nécessite la mise en place d’une *IGC* (*Infrastructure de Gestion des Clefs*) [RFC3647] [RFC3280] et des certificats d’identité.

La cryptographie à clefs publiques [DIF] est perçue actuellement comme l'élément de base de toute solution robuste de sécurité et des applications exigeant des services de sécurité, notamment d’authentification forte, d’identification, de non-répudiation, et d’horodatage [BUL] [HAB] dans un réseau ouvert (le réseau Internet par exemple). Son utilisation par le plus grand nombre d’application nécessite non seulement, sur le plan technique, une infrastructure permettant la publication et la gestion des clefs publiques, mais aussi sur le plan des usages, la définition de services en phase avec la demande et les exigences juridiques.

Dans ce chapitre, nous présentons de manière générale les *IGC*s et les certificats numériques. Ensuite nous focalisons l'étude sur les certificats d’identités X.509v3 et leurs limites. Ce chapitre est ainsi structuré en trois parties. Dans la première partie, nous introduisons les certificats numériques. La deuxième partie traite l’*IGC* (définition, composants). Dans une troisième partie, nous illustrons tous les aspects liés aux certificats d’identité X.509 et leur usage, et notamment à leurs limites.

2.2 Les certificats numériques

L’utilisation des clefs publiques repose sur la confiance en l’entité d’origine. En effet, lorsqu’une liaison sécurisée est établie entre deux utilisateurs, rien ne prouve que l’un d’entre eux n’est pas un imposteur. Les attaques de type "*man in the middle*" [COH] ou "*maskerading*" [HAY] consistent à se faire passer pour une entité connue en usurpant son identité. Pour pouvoir utiliser une clef publique avec sécurité, il faut donc que le récepteur puisse savoir à qui appartient cette clef publique et à quoi elle sert. Pour cela, on utilise un système de certificats.

Par définition, un certificat est un document établissant un lien entre une clef et un sujet ou un privilège [RF3280] [RFC3281]. Cette liaison dépend du contexte et du type de certificat. Un certificat peut servir à établir un ou plusieurs faits dont la confirmation de l'identité d'une personne, de l'identification d'une société, d'une association ou d'un État, de l'exactitude d'un identifiant d'un document ou d'un autre objet, de l'existence de certains attributs d'une personne, d'un document ou d'un autre objet ou encore du lien entre eux et un dispositif d'identification ou de localisation tangible ou logique. Il s'agit d'une preuve assurant l'association entre la clef publique et les informations concernant l'identité de son possesseur, qu'il s'agisse d'un individu, d'un ordinateur, d'un service informatique, etc. Cette association est garantie par la signature numérique d'une autorité de certification. Ainsi, il suffit de faire confiance à cette autorité de certification pour être assuré que la clef publique incluse dans le certificat appartienne à la personne qui y est décrite.

Il existe plusieurs formats de certificats qui ont été définis : *SPKI* [RFC2692] [RFC2693], Certificat d'identité X.509 [RF3280], Certificat d'Attributs [RFC3281], PGP [RFC1991].

2.3 Infrastructure de Gestion des Clefs

2.3.1 Définition

PKI [RFC3647] est un acronyme anglo-saxon signifiant "*Public Key Infrastructure*", souvent traduit en français par *ICP* (*Infrastructure à Clefs Publiques*) ou *IGC* (*Infrastructure de Gestion des Clefs*). Nous adoptons dans cette thèse le terme *IGC*.

Cette infrastructure est composée d'un ensemble de moyens matériels, de logiciels, de composants cryptographiques, mis en œuvre par des personnes. Ces moyens sont combinés par des politiques, des pratiques et des procédures requises, qui permettent de créer, gérer, conserver, distribuer et révoquer des certificats basés sur la cryptographie asymétrique [EYRAUT].

Ce système de gestion des clefs publiques (*IGC*) permet de gérer des listes importantes de clefs publiques et d'en assurer la fiabilité pour des entités, généralement dans un réseau. L'*IGC* offre un cadre global permettant d'installer les éléments de sécurité tels que la confidentialité, l'authentification, l'intégrité et la non-répudiation, tant au sein de l'entreprise que lors d'échanges d'information avec l'extérieur.

L'utilisation d'une infrastructure de gestion des clefs répond aux besoins des organisations qui utilisent des services de sécurité basés sur les technologies de clefs asymétriques (voir annexe A).

2.3.2 Les composants d'*IGC*

Une *IGC* est composée de trois éléments principaux qui sont la CA (*Certificate Authority* ou Autorité de Certification), la RA (*Register Authority* ou Autorité d'Enregistrement) et d'un annuaire où sont stockées les informations (voir figure 2-1).

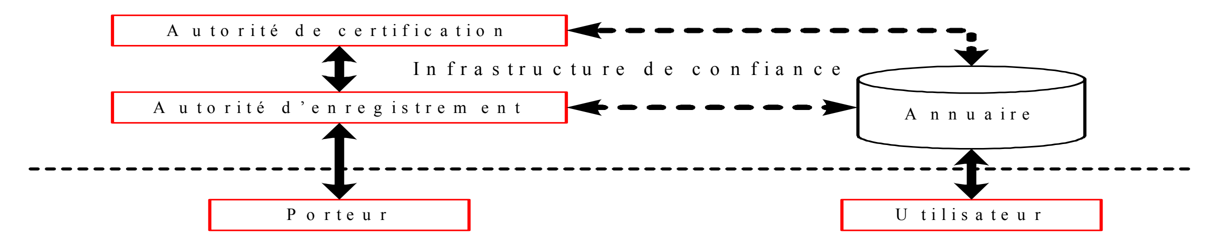


Figure 2-1 Découpage des acteurs *IGC*

2.3.2.1 Autorité de Certification

Une autorité de certification CA (*Certificate Authority)* est chargée d'émettre et de gérer des certificats numériques. Elle est l'organe central de l'*IGC* et est responsable de l'ensemble du processus de certification et de la validation des certificats émis.

Une autorité de certification doit définir une politique de certification qui établira l'ensemble des règles de vérification, de stockage et de confidentialité des données appartenant à un certificat, ainsi que la sécurité du stockage de sa propre clef privée nécessaire à la signature des certificats. La politique de certification appartient à une *IGC*.

L'autorité de certification peut définir plusieurs politiques de certification en fonction du mode d'enregistrement et de l'usage du certificat. Elle distingue alors des classes de certificats et définit en même temps les conditions et le niveau de sa responsabilité.

Lorsque l'autorité d'enregistrement est assurée de l'identité et des droits du demandeur, de manière plus ou moins poussée selon la garantie associée, l'autorité de certification prend la décision d'émettre un certificat adapté. C'est cette dernière qui va authentifier le certificat des usagers en le signant. Les données contenues dans le certificat sont ainsi protégées en intégrité. Elle associe le certificat à l'identité d'une entité (équipement ou usager). Par ailleurs, elle peut elle-même générer la paire de clefs publique/privé ou laisser cette tâche à l'usager. C'est la tâche la plus sensible d'une *IGC* car c'est avec sa clef privée que l'ensemble des certificats soumis ou générés vont être signés. La CA est dotée d'un certificat qu'elle peut signer elle-même et qu'on qualifie d'auto-signé, ou qui peut être signé par une CA de plus haut niveau. Ce système de confiance est appelé modèle de confiance hiérarchique. D'autre part, pour des raisons de facteurs d'échelle, et dans la plupart des cas, il n'est pas possible qu'une seule autorité de certification s'occupe de toutes les entités. Il va falloir ainsi employer plusieurs méthodes ou modèles de confiance pour mettre en relation les différentes autorités de certification (les modèles de confiance sont présentés à l'annexe B). Le certificat du CA doit être diffusé d'une manière sûre pour permettre à l'ensemble des entités de l'authentifier comme "signé" par ce CA.

L'autorité de certification est responsable non seulement de l'émission des certificats, mais aussi de leur gestion durant tout leur cycle de vie, et en particulier s'il y a besoin, de leur révocation. Pour les prestations techniques, elle s'appuie sur un Opérateur de Certification, qu'il soit interne ou externe, dont elle approuve et audite les moyens et procédures.

Le choix du modèle de confiance est l'un des points capitaux du déploiement d'une *IGC* au sein d'une communauté. La notion de confiance est une notion complexe. Elle est cependant définie dans la recommandation X.509 comme telle: une entité A fait confiance à une entité B lorsque A suppose que B se comportera tel que A l'attend [RFC3647]. Cette notion est donc difficile à appréhender d'un point de vue technique. il s'agit avant tout d'une relation de type humain. Il est important de noter que, dans la plupart des modèles de confiance, une règle s'applique à la notion de confiance, c'est la transitivité. Si A fait confiance à B et B fait confiance à C, alors A fait confiance à C. Ceci est la base des relations entre les différentes autorités de certifications.

2.3.2.2 Autorité d'Enregistrement

Bien que la fonction d'enregistrement puisse être implémentée directement avec les composants de la CA, il peut être intéressant de l'utiliser en tant que composant distinct. On l'appelle alors autorité d'enregistrement RA (*Registration Authority*). Par exemple, lorsque le nombre d'entités finales dans un domaine *IGC* augmente et/ou si les entités sont dispersées géographiquement, la notion d'enregistrement centralisé devient alors problématique. Un déploiement judicieux de multiples RA (que l'on appelle parfois autorité d'enregistrement locale, *Local Registration Authority*, LRA) aide à résoudre ce problème. Le premier objectif d'une RA est de décharger la CA de certaines fonctions.

L'autorité d'enregistrement prend en charge l'arrivée d'une nouvelle entité au sein de la communauté et sur le réseau. La RA est le « guichet » où le certificat est demandé. Elle représente le point de contact entre l'utilisateur et l'autorité de certification. Elle joue plusieurs rôles, notamment la gestion des requêtes soumises à la CA. La RA prend en charge de vérifier l'identité du demandeur du certificat. La vérification se fait selon des critères précis définis par la CA. La première opération à effectuer lors de l'arrivée de ce demandeur est la création de son profil, c'est-à-dire l'enregistrement d'informations le concernant.

Dans le cas où un utilisateur fait une demande de certificat, la RA vérifie les informations qu'il a données, et les modifie si besoin. Ensuite, elle génère une requête de certificat pour la CA, cette requête respecte le format PKCS#10 [RFC2986]. La RA effectue aussi d'autres vérifications suivant la politique de certification mise en œuvre. Une fois ces preuves obtenues, la demande est acceptée.

La communication entre la RA et la CA se fait de manière sécurisée. Ces autorités utilisent le cryptage asymétrique pour correspondre entre elles. Chacune possède et utilise la clef publique de l'autre.

Les fonctions implémentées dans une RA varient. Néanmoins, on retrouve souvent une ou plusieurs des fonctions suivantes :

- Etablir et confirmer l'identité d'un individu lors du processus d'initialisation de demande de certification d'une entité.
- Distribuer des clefs secrètes partagées aux utilisateurs finaux pour l'authentification, au cours d'un processus d'initialisation en ligne.

Parallèlement, l'autorité de certification devait générer une paire de clef publique/privée pour cet utilisateur, qui pourra ensuite être associée aux informations précédentes au sein d'un certificat. Celui-ci sera ensuite publié par un annuaire ou alors envoyé de façon sécurisée (ou carrément par d'autres moyens que le réseau) aux personnes qui en ont besoin. Lors de l'opération de génération de la paire de clefs, on peut, dans certains cas, en faire une copie de sauvegarde.

L'autorité d'enregistrement gére donc le processus d'arrivée des clients. Son rôle principal est de recevoir les informations sur l'utilisateur. Ici apparaît une contrainte importante qui est la capacité d'authentifier l'utilisateur lors de l'accès à l'autorité d'enregistrement. Tout dépend en fait de la politique de certification adoptée par l'entreprise. En effet, si les informations que les certificats protègent sont très importantes, la procédure d'enregistrement demande la présence physique de l'utilisateur. Dans le cas contraire, on pourra réaliser une procédure d'enregistrement *on-line*, à travers un formulaire; par exemple, en prenant le soin d'authentifier l'utilisateur par un mot de passe ou toute autre méthode. Là encore, l'ensemble de ces modalités est défini par la politique de certification de l'entreprise. La définition de cette politique est l'une des grosses difficultés du déploiement de la *IGC*.

La génération de la paire de clefs peut être effectuée à différents niveaux. Soit elle est externe, la paire de clefs étant une information que l'autorité d'enregistrement récupère. Soit elle est effectuée par l'autorité d'enregistrement, ou encore elle peut être effectuée par l'autorité de certification. Les communications entre le client d'enregistrement, l'autorité d'enregistrement et l'autorité de certification sont bien sûr sécurisées. Cette opération est généralement très coûteuse en terme de puissance de calcul. On est donc dans la plupart des cas confiée à une carte matérielle dédiée. Cette remarque est valable pour de nombreux composants d'une *IGC* car les opérations de cryptographie sont généralement coûteuses et consomment d'autant plus de puissance de calcul que les longueurs des clefs sont longues.

L'opération de certification a déjà été détaillée. Cependant, nous n'avons pas encore parlé du problème de la distribution des certificats. Là encore, le mode de distribution est fixé par la politique de certification de l'entreprise. Le choix d'un moyen de distribution d'un certificat dépend avant tout de celui à qui appartient le certificat et quel en sera l'usage. S'il s'agit du certificat d'un utilisateur banal ne nécessitant pas de précautions particulières, l'utilisateur une fois enregistré, pourra le recevoir

directement sur son poste par une communication sécurisée sur le réseau (il existe d'ailleurs des protocoles qui permettent le retrait des certificats de façon sécurisée auprès des autorités de certification). Si la clef protégée par le certificat est bien plus importante, on pourra envisager d'autres moyens de distribution comme un transport physique de la clef ou autres.

Comme nous l'avons suggéré, dans certains cas, l'Autorité d'Enregistrement peut procéder à une copie de sauvegarde de la clef et du certificat lors de leurs générations. Cette opération peut être d'une importance capitale, car il est hors de question d'imaginer la possibilité qu'une entreprise perde une partie de ses informations à la suite de la perte ou de la dégradation d'une clef. La zone de stockage des clefs et certificats de sauvegarde doit être particulièrement protégée, car elle est le moyen de percer tout le système de sécurité de l'entreprise. Notons aussi que les certificats ont une durée de vie finie. Dans la plupart des cas, la paire de clef est régénérée lorsque la période de validité expire, il est donc très important d'automatiser le processus de sauvegarde de façon à toujours posséder les clefs nécessaires au décryptage de l'ensemble des informations et de gérer la période de transition. De même, certaines clefs possèdent des dates d'expiration et il faudra gérer ceci par rapport au certificat qui les représente.

2.3.2.3 Annuaire

L'annuaire est le lieu où sont stockés les certificats de clef publique (*certificats d'identité*) ainsi que les listes de révocations des certificats CRL (*Certificate Revocation List*) [RFC3280] [RFC3279]. L'annuaire est donc un composant de l'*IGC* qui doit être capable non seulement de stocker ces informations et de permettre un accès facile, mais aussi un accès réglementé. En fait, le terme annuaire est un terme générique pour désigner un ensemble de technologies parmi lesquelles on peut citer LDAP (*Lightweight Directory Access Protocol*) [RFC2251], les recommandations X.500 [RFC1279], les serveurs OCSP (*Online Certificate Status Protocol*) [RFC2560], *DNS* (*Domain Name System*) [RFC1591] (utilisant les certificats et les révocations), les serveurs HTTP (*Hypertext Transfer Protocol*) [RFC2616], les serveurs FTP (*File Transfer Protocol*) [RFC959], les bases de données, etc. Cependant, le protocole consacré en matière d'annuaire (*repository* en anglais) est LDAP. Généralement, les accès en lecture à l'annuaire sont totalement libres, mais sont par contre réglementés en ce qui concerne le dépôt d'informations.

L'utilisation d'un annuaire présente, par contre, un certain nombre d'inconvénients dont la génération d'un trafic réseau important, dû au retrait et au dépôt des certificats et des listes de révocation. L'autre inconvénient est de savoir s'il est simple de déployer une solution capable de supporter la charge d'un ensemble conséquent d'utilisateurs. Une autre question centrale reste celle du contrôle des accès. Celle-ci est peu préoccupante dans le cas où l'annuaire a un usage limité au domaine de l'entreprise. D'autre part, la question peut s'avérer plus corsée si l'annuaire peut être utilisé par une autre entreprise. Il faut donc faire attention à ce que la liberté d'accès à l'annuaire ne soit pas un problème de sécurité et laisser éventuellement des accès à des informations sensibles.

2.4 Les certificats X.509

Le format des certificats le plus utilisé aujourd'hui dans le cadre d'une *IGC* est le format normalisé par le standard X.509 dans sa version trois [RFC3280].

Parce que la nouvelle version de la norme X.509 définit maintenant deux types de certificats, le terme certificat X.509, depuis longtemps en usage, désigne désormais deux types d'objets distincts:

- Le certificat de clef publique: il attache ensemble une valeur de clef publique et une identité (Nom de l'utilisateur auquel appartient le certifficat, "*subject X.500 name*") (voir figure 2-2). Il est analogue à un passeport (lequel suppose une vérification préalable d'identité).

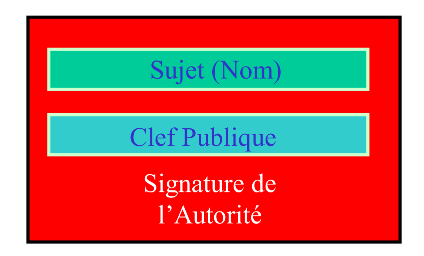


Figure 2-2 Certificat d'identité X.509

- Le certificat d'attributs (voir figure 2-3) : il attache ensemble une identité et des attributs (Droits de l'identité) servant à véhiculer des autorisations. Il est analogue à un visa (lequel suppose une vérification du passeport).

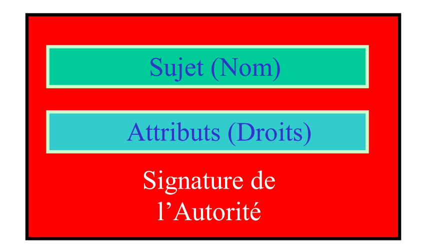


Figure 2-3 Certificat d'Attributs X.509

L'ITU (*International Telecommunication Union*) [ITUT] et l'ISO (*International Organisation for Standarisation*) [ISO] ont publié la première version du standard X.509 [ITUT-X509-97], qui a été adoptée par l'*IETF* (*International Enginering Task Force*) [IETF]. À l'origine conçue en 1988, le certificat X.509 est passé par trois versions principales (voir figure 2-4) pour arriver à sa forme actuelle.

La première version de X.509 avait pour but de décrire les méthodes d'authentification et de contrôle d'accès aux annuaires X.500. Cette version a ensuite été reprise et modifiée dans deux versions successives.

Lors de la révision de la norme X.500 en 1993, la deuxième version a ajouté deux champs pour supporter les contrôles d'accès aux répertoires, qui sont les champs "*issuer*" et "*subject identifier*". Enfin pour permettre l'ajout d'autres informations (attributs ou privilèges) dans un certificat, la troisième version de X.509 a introduit le concept d'extension qui sont optionnelles. Il s'agit de blocs d'informations permettant de stocker des données supplémentaires. Ces extensions introduites permettent de spécifier un certain nombre d'informations en fonction de l'usage prévu d'un certificat.

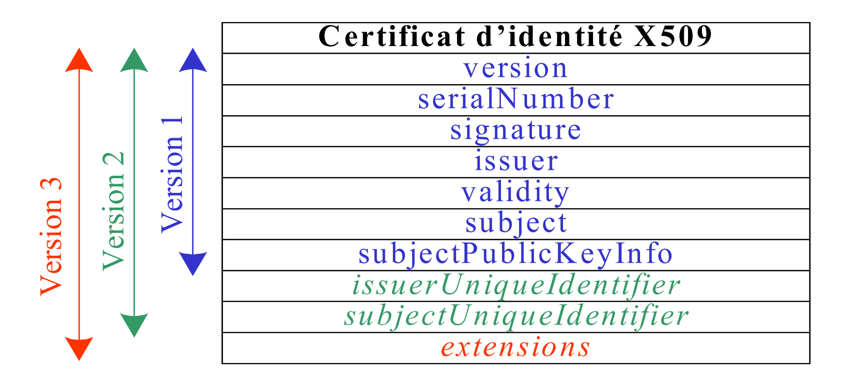


Figure 2-4 Différentes versions de certificat d'identité X.509

2.4.1 Spécification ASN.1 d'un certificat X.509

La définition du certificat dans le standard X.509 utilise la notation *ASN.1* (*Abstract Syntax Notation number One*) [ITUT-X208-88] (voir figure 2-5) pour spécifier les champs du certificat.

ASN.1 est une norme internationale dont la vocation primaire est la spécification de données utilisées dans les protocoles de communication. Il s'agit d'un langage informatique à la fois puissant et complexe. Ses traits ont été conçus pour que le langage modélise efficacement la communication entre systèmes hétérogènes.

ASN.1 a pour but de décrire la structure des données d'une application répartie, indépendamment de toute considération logicielle et matérielle.

ASN.1 ne décrit pas le contenu, ni la signification des données, mais uniquement la manière dont elles sont spécifiées et codées (pas de sémantique). Cette notation qui est utilisée pour décrire les messages échangés entre des applications communicantes offre un haut niveau de description qui évite aux personnes spécifiant les protocoles de se préoccuper de la disposition des bits ou des octets dans les transferts de données. Des ensembles de règles de codage normalisés sont étroitement associés à *ASN.1*. Ils décrivent la disposition des bits ou des octets lorsque les messages circulent entre les applications communicantes. Le codage BER (*Basic Encoding Rules*) [ITUT-X209-88] [ITUT-X690-97], ou sa dérivée DER (*Distinguished Encoding Rules*) [ITUT-X690-97], sont utilisés pour représenter concrètement les données *ASN.1* en tableaux d'octets. Les codages BER et DER sont indépendants des plates-formes utilisées. Ceci permet d'échanger des données sans problème. Le passage de cette définition abstraite de *ASN.1* vers un certificat numérique X.509 se fait en appliquant les règles d'encodage DER. Les données sont transmises en format binaire, ce qui offre de bonnes qualités de compression et de vitesse de transmission.

```
Certificate ::=    SIGNED { SEQUENCE {
    version                   [0]    Version DEFAULT v1,
    serialNumber                     CertificateSerialNumber,
    signature                        AlgorithmIdentifier,
    issuer                           Name,
    validity                         Validity,
    subject                          Name,
    subjectPublicKeyInfo             SubjectPublicKeyInfo,
    issuerUniqueIdentifier    [1]    IMPLICIT UniqueIdentifier OPTIONAL,
                              -- if present, version shall be v2 or v3
    subjectUniqueIdentifier   [2]    IMPLICIT UniqueIdentifier OPTIONAL,
                              -- if present, version shall be v2 or v3
     extensions               [3]    Extensions OPTIONAL
                              -- If present, version shall be v3 -- } }
```

Figure 2-5 Le formalisme *ASN.1* du certificat d'identité X.509

2.4.2 Contenu d'un certificat X.509v 3

Un certificat X.509v3 contient les données énumérées dans la figure 2-6. Ce format regroupe un certain nombre de champs "standards" et de champs dits "d'extension", qui permettent de personnaliser les certificats en fonction de leur usage.

Néanmoins, une partie des extensions est "normalisée" par X.509v3 et notamment celles permettant de spécifier un certain nombre d'informations en fonction de l'usage prévu d'un certificat. Notons que certains de ces champs sont facultatifs.

Nous décrivons par la suite le contenu des champs standards, puis des extensions les plus couramment utilisées.

2.4.2.1 Les champs « standards »

Les significations des champs que l'on trouve systématiquement dans un certificat X.509v3 sont les suivantes :

- *Certificate format version* : ce champ indique à quelle version de X.509 correspond ce certificat.
- *Certificate serial number* : un numéro de série unique, propre à chaque autorité de certification, qui l'identifie de façon unique. C'est ce numéro de série qui sera posté dans la liste de révocation en cas de révocation.
- *Signature algorithm identifier for CA* : elle désigne le procédé utilisé par l'autorité de certification pour signer le certificat : (norme ISO). Il s'agit d'un algorithme asymétrique et d'une fonction de condensation.
- *Issuer X.500 name* : ce champs spécifie le nom (*Distinguished Name*) dans la norme X.500 de l'autorité de certification qui a émis le certificat.
- *Validity period* : ce champs spécifie les dates de début et de fin de validité du certificat.
- *Subject X.500 name* : ce champs spécifie le le nom (*Distinguished Name*) du propriétaire de ce certificat.
- *Subject public key information* : ce champs contient la clef publique du détenteur du certificat ainsi que les algorithmes avec lesquels elle doit être utilisée.
- *Issuer unique identifier (optionnel)* : ce champs optionnel qui est apparu en v2, permet de donner une seconde identification au *Issuer X.500 name* (la CA) dans le cas où celui-ci a un nom (*Distinguished Name*) commun avec une autre CA.
- *Subject unique identifier (optionnel)* : ce champs optionnel qui est apparu en v2, permet de donner une seconde identification au *Subject X.500 name* (l'utilisateur) dans le cas où celui-ci a un nom (*Distinguished Name*) en commun avec un ou plusieurs autres utilisateurs.
- *CA signature* : c'est la signature de l'autorité de certification (CA). Cette signature est effectuée en passant l'ensemble du certificat au travers d'une fonction de hachage, puis en chiffrant le résultat à l'aide de la clef privée de l'autorité de certification.

Version du certificat (certificate format version)
Numéro de série du certificat (certificate serial number)
Description de l'algorithme de signature de l'AC (signature algorithm identifier for CA)
Nom de l'AC qui a généré le certificat (issuer X.500 name)
Période de validité (validity period)
Nom de l'utilisateur auquel appartient le certificat (subject X.500 name)
Clef publique (subject public key)
Description de l'algorithme à utiliser avec la clef publique (subject public key information)
Identification possible de l'AC « optionnel »
Identification possible de l'utilisateur (subject unique identifier) « optionnel »
Extensions « optionnel »
Signature de l'AC (CA signature)

Figure 2-6 Contenu d'un certificat d'identité X.509v3

2.4.2.2 Les champs extensions « optionnels »

Les champs extensions des certificats X.509v3 fournissent les moyens d'associer des informations supplémentaires pour le champ *sujet* (Nom de l'utilisateur auquel appartient le certificat) et/ou pour la *clef publique* du certificat. Ils permettent également de gérer la hiérarchisation des certificats. Ce mécanisme permet aussi de définir des extensions propres à une politique ou une communauté de certification.

Chaque extension est une combinaison d'identifiant (*Object Identifier, OID*) et de structure *ASN.1*. Chaque OID ne doit apparaître qu'une seule fois.

Chaque extension est constituée de trois champs (voir figure 2-7).

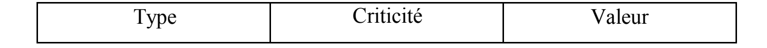


Figure 2-7 Extension de certificats X.509v3

Le *Type* représente l'identifiant de l'extension donné par la norme X.509. La *Criticité* indique si le champ doit être considéré comme critique par l'application, et enfin le champ *Valeur* est la valeur proprement dite de l'extension considérée.

Le caractère de criticité devrait se traiter de la façon suivante, selon que l'extension est critique ou pas :

➢ Si l'extension n'est pas critique alors :

- Soit l'application ne sait pas la traiter, et alors l'extension est abandonnée mais le certificat est accepté ;
- Soit l'application sait la traiter, et alors :
  – Si l'extension est conforme à l'usage que l'application veut en faire, l'extension est traitée.
  – Si l'extension n'est pas conforme à l'usage que l'application veut en faire, l'extension est abandonnée, mais le certificat est accepté.

➢ Si l'extension est critique, alors :

- Soit l'application ne sait pas la traiter, et le certificat est alors rejeté.
- Soit l'application sait la traiter, et alors :
  – Si l'extension est conforme à l'usage que l'application veut en faire, l'extension est traitée.
  – Si l'extension n'est pas conforme à l'usage que l'application veut en faire, le certificat est rejeté.

On peut grouper les extensions en quatre types :

1. **Informations sur les clefs**

Ce groupe d'extensions, qui renseigne sur l'utilisation qui doit être faite de la clef publique et du certificat, est constitué des champs suivants :

- *Authority Key Identifier* : ce champ identifie de façon unique la paire de clefs utilisée par la CA pour signer le certificat. Il est utilisé pour faciliter le processus de vérification de la signature du certificat dans le cas où la CA aurait utilisé plusieurs clefs depuis sa mise en œuvre.

L'extension *Authority Key Identifier* accepte deux options: *Keyid* et *issuer*. Toutes les deux peuvent prendre la valeur facultative *always*. Cette extension ne doit pas être marquée *critical* [RFC3280].

Certificat	Extension AuthorityKeyIdentifier	RFC
Root CA	-	MAY
CA Interm.	AuthorityKeyIdentifier = Keyid : always, issuer : always	MUST
Serveur	AuthorityKeyIdentifier = Keyid : always	MUST
Utilisateur	AuthorityKeyIdentifier = Keyid : always	MUST

Tableau 2-1 Extension AuthorityKeyIdentifier

L'option *Keyid* tente de copier la valeur du *subjectKeyIdentifier* du certificat parent. Si seule l'option *Keyid* est positionnée et contient la valeur *always* et si la tentative échoue, une erreur est alors retournée.

L'option *issuer* copie le champ *subject* et le *numéro de série* du certificat signataire. Cette option est appliquée seulement si elle contient la valeur *always* et/ou si l'option *Keyid* échoue.

- *Subject Key Identifier* : ce champ identifie de façon unique la clef publique qui est contenue dans le certificat. On y recourt lorsqu'un utilisateur possède un historique de clefs de chiffrement.

Pour les certificats d'entités finales (Utilisateur, Serveur, etc.), l'extension *Subject Key Identifier* fournit un moyen d'identifier les certificats contenant la clef publique utilisée dans une application. Dans le cas où une entité finale aurait obtenu plusieurs certificats, notamment de plusieurs CA différentes, l'extension *Subject Key Identifier* fournit un moyen d'identifier rapidement l'ensemble de certificats contenant une clef publique donnée.

Cette extension ne doit pas être marquée *critical* [RFC 3280].

Certificat	Extension KeyUsage	RFC
Root CA	KeyUsage = Critical, KeyCertSign, CRLSign	MUST
CA Interm.	KeyUsage = Critical, KeyCertSign, CRLSign	MUST
Serveur	KeyUsage = DigitalSignature, KeyEncipherement	-
Utilisateur	KeyUsage = DigitalSignature, nonRepudiation, KeyEncipherement	-

Tableau 2-2 Extension KeyUsage

- *Key Usage* : ce champ définit les utilisations de la clef contenue dans le certificat (par exemple, chiffrement, signature de données, signature de certifcats). Les paramètres de l'extension *Key Usage* sont des *flags* qui peuvent ou non être positionnés. Le champ *Key usage* peut avoir les valeurs présentées dans le tableau ci-dessous:

Usage de la clef	Key Usage	Combinaisons Valides				
		Fonction de signature	Fonction de certification (signature de certificats/CRL).	Fonction de chiffrement de clef	Fonction de chiffrement de données	Fonction de négociation de clef
Signature numérique	DigitalSignature	X				
Chiffrement de clef	KeyEnciphrement			X		
Chiffrement de données	DataEnciphrement				X	
Non Répudiation	nonRepudiation	X				
Négociation de clefs	KeyAgreement					X
Clef de signature de certificat	KeyCertSign		X			
Signature de CRL	CRLSign		X			
Chiffrement seul	encipherOnly					
Déchiffrement seul	decipherOnly					

Tableau 2-3 Les valeurs du champ Key Usage

La description de ces paramètres est présentée dans le tableau ci-dessous:

Paramètres	Description de l'utilisation
DigitalSignature	Pour les vérifications de signatures numériques qui ont des buts autres que ceux décrits pour les paramètres nonRepudiation, KeyCertSign, CRLSign.
KeyEncipherement	Pour chiffrer des clefs ou toute autre information de sécurité, par exemple, pour le transport de clefs.
DataEnciphrement	Pour chiffrer des données d'utilisateur, et non des clefs ni toute autre information de sécurité.
NonRepudiation	Pour les vérifications de signatures numériques utilisées avec un service de non-répudiation. Ce paramètre exclut les cas des paramètres KeyCertSign et CRLSign.
KeyAgreement	Pour utiliser la clef publique comme « clef d'accord ».
KeyCertSign	Pour vérifier une signature de certificats par un certificat «Root» CA.
CRLSign	Pour vérifier une signature de CRL par un certificat «Root» CA.
EncipherOnly	Pour le chiffrage de données uniquement. Il doit être utilisé avec le paramètre KeyAgreement uniquement.
DecipherOnly	Pour le déchiffrage de données uniquement. Il doit être utilisé avec le paramètre KeyAgreement uniquement.

Tableau 2-4 Description des paramètres du champ Key Usage

- *Private Key Usage Period* : ce champ donne la date d'expiration de la clef privée qui est associée à la clef publique contenue dans le certificat. Il est généralement utilisé pour la clef privée de signature dans le cas où sa période d'utilisation serait différente de la période de validité du certificat.
- *CRL Distribution Points* : cette extension définit l'emplacement de la liste de révocation des certificats CRL (*Certificate Revocation List*) d'une autorité de certification. Cette extension ne doit pas être marquée *critical* [RFC3280].

2. Informations sur l'utilisation du certificat

Ce groupe d'extensions permet de spécifier l'utilisation des certificats en accord avec la politique définie dans l'*IGC*.

Ces extensions sont représentées dans les deux champs suivants :

- *Certificate Policies* : ce champ spécifie la Politique de Certification (PC) qui a présidé à l'émission du certificat. Les Politiques de Certification sont représentées par des *Object IDentifier* (*OID*). Ces *OID* sont enregistrés au niveau international et leurs utilisations sont standardisées.

  Si ce champ est marqué comme étant critique, l'*IGC* impose que le certificat soit utilisé conformément à la PC. Dans le cas contraire, l'information est indicative. Ensuite, libre à l'application cliente de respecter ou pas la criticité.
- *Policy Mappings* : ce champ ne concerne que les co-certificats (le certificat émis par une CA pour certifier la clef publique d'une autre CA). Il permet d'associer la PC d'une CA qui émet le certificat à la PC indiquée dans le co-certificat. S'il est défini comme critique, il permet aux applications qui vérifient un certificat appartenant à une chaîne de certification, de s'assurer qu'une PC qui peut être acceptée s'applique à tous les certificats.

3. Attributs des utilisateurs et des autorités de certification

Ce groupe d'extensions permet de mieux spécifier l'identification des utilisateurs et des certificats.

- *Subject Alternative Name* : ce champ spécifie une ou plusieurs informations sur le propriétaire du certificat. Les valeurs autorisées sont : une adresse e-mail, un nom de *DNS*, une adresse IP [RFC791], une adresse e-mail X.400 [RFC2162], un nom EDI [RFC1767], une URL [RFC1738],

un nom défini par une OID. L'extension *Subject Alternative Name* ne devrait jamais être marquée *critical* [RFC3280].

- *Issuer Alternative Name* : ce champ permet de donner un nom spécifique à une CA.

Certificat	Extension SubjectAltName et IssuerAltName
Root CA	-
CA Interm.	-
Serveur	-
Utilisateur	SubjectAltName = email : copy

Tableau 2-5 Extension SubjectAltName et IssuerAltName

4. Contraintes sur la co-certification

En cas de co-certification, les extensions de ce groupe permettent de limiter et de contrôler les indices de confiance envers d'autres CA.

- *Basic Constraints* : ce champ indique si le sujet du certificat est un utilisateur final ou s'il est une autorité de certification. Dans ce dernier cas, le certificat est un co-certificat.

  Cette extension indique également le nombre maximum de niveaux de certificats intermédiaires qui peuvent être issus de cette autorité de certification. On définit alors une «distance de certification» "*pathlen*" qui spécifie jusqu'où doit remonter une application qui veut vérifier un certificat en consultant sa liste de révocation des certificats CRL, et jusqu'où est étendue la confiance dans la chaîne. Si cette distance est par exemple à 1, les utilisateurs ne peuvent que vérifier les certificats émis par la CA définie dans le co-certificat.

  Cette extension doit être marquée *critical* pour les certificats CA (c'est-à-dire CA : true) dont la clef publique sert à valider la signature d'un certificat. Elle peut être marquée *critical* ou non pour les autres certificats [RFC3280].

Certificat	Extension BasicConstraints	RFC
Root CA	BasicConstraints = critical, CA : true	MUST
CA Interm.	BasicConstraints = critical, CA : true	MUST
Serveur	BasicConstraints = critical, CA : false	SCHOULD
Utilisateur	BasicConstraints = critical, CA : false	SCHOULD

Tableau 2-6 Extension BasicConstraints

  Une autorité de certification avec un "*pathlen*" positionné à zéro ne pourra être employée que pour signer des certificats d'entités finales et non pour signer des certificats CA Intermédiaire.

- *Name Constraints* : ce champ n'est utilisé que dans les co-certificats et permet aux administrateurs de restreindre les domaines de confiance dans un domaine de co-certification.

- *Policy Constraints* : ce champ s'applique aux co-certificats, et permet de spécifier les politiques de certifications acceptables pour les certificats dépendants du co-certificat.

2.4.3 Limites des certificats X.509v3

Les extensions de certificats X.509v3 sont des champs qui permettent de rendre l'utilisation des certificats beaucoup plus flexible. Néanmoins, elles rendent bien souvent les applications traitant ces certificats incompatibles entre elles (voir paragraphe 2.4.3.1) [SSGRR03] [SETIT03].

Ainsi, la prolifération de ces extensions génère des problèmes dans les domaines d'interopérabilité et de révocation.

2.4.3.1 Interopérabilité :

Une extension dans un certificat peut être qualifiée de critique ou non. Le fait qu'une extension soit critique rend obligatoire la conformité du certificat aux informations contenues dans l'extension. Si une application traitant un certificat ne reconnaît pas une extension contenue dans ce certificat, et elle est marquée comme critique, ce certificat doit être déclaré invalide par l'application. Dans ce cas, ce champ crée des problèmes d'interopérabilité, ce qui annule les avantages offerts par les extensions en premier lieu. En particulier, les extensions *key Usage* et *Basic Constraint* doivent être marquées critiques.

Le bon fonctionnement de l'échange de données spécifiées en *ASN.1* entre deux machines dépend de la compatibilité de leur fonction de codage et de décodage. Si elles ne sont pas réciproques l'une avec l'autre et ne supportent pas les mêmes champs, cela peut être source de non interopérabilité entre ces machines. Dans le cas des certificats, les problèmes peuvent notamment survenir pour les extensions qui sont optionnelles et qui ne sont pas supportées par tous les compilateurs *ASN.1*, bien qu'elles puissent être critiques.

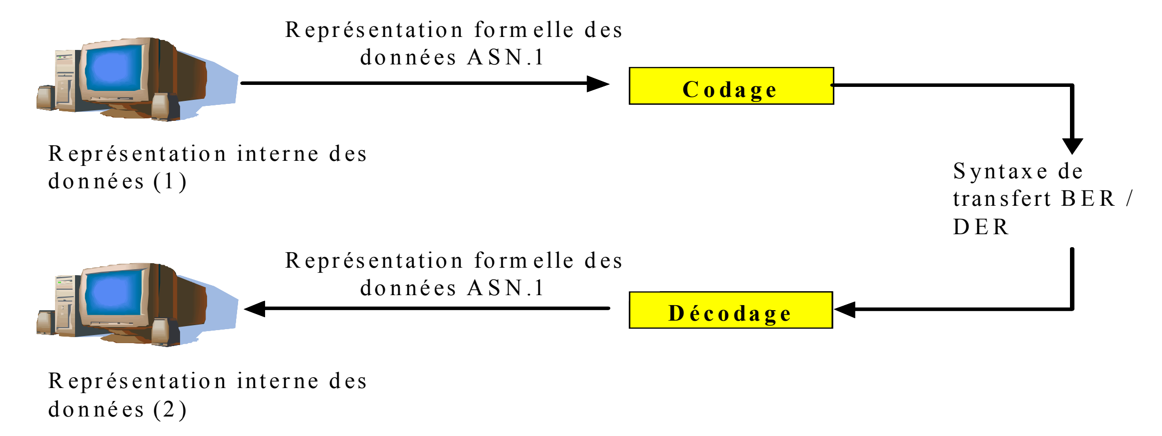


Figure 2-8 Chaîne de codage et décodage des données en *ASN.1*

2.4.3.2 Révocation :

L'agrégation des attributs a comme conséquence le raccourcissement de la durée de vie du certificat, puisque chaque attribut a sa propre durée de vie. Ainsi, du fait que les attributs possèdent des durées de vie courtes et que leur contenu change plus souvent que l'identité, la fréquence de révocation du certificat augmente.

Une solution apparaît pour résoudre et simplifier le problème de révocation. Elle consiste à partager un certificat X.509 en deux certificats : un certificat d'identité qui contient des informations sur l'identité et un certificat d'attributs qui contient des informations sur l'attribut (une étude sur les certificats d'attributs sera détaillée dans le chapitre suivant). Si les certificats d'attributs ont une durée de vie très courte, ils n'auraient pas besoin d'être révoqués.

2.5 Conclusion

Dans ce chapitre, nous avons d'abord présenté une vision globale des Infrastructures de Gestions des Clefs et de leurs composants. Les *IGC*s utilisent la norme X509v3 qui offre une garantie au niveau de l'authentification des entités dans le réseau global. Cette norme et les expérimentations associées ne traitent pas directement le problème du contrôle d'accès. La norme X509 définit des certificats d'identité qui prouvent la relation entre la clef publique et le nom de l'entité (*Distinguished Name*) à qui appartient cette clef publique.

Nous avons introduit les certificats d'identité X.509, en particulier la version 3, ainsi que leurs limites. Cette étude était nécessaire pour montrer la lourdeur de la structure des certificats d'identité X.509 et de sa mise en œuvre. Les certificats peuvent rendre de nombreux services comme l'authentification, la non répudiation, l'intégrité et la confidentialité. Néanmoins, il faut émettre quelques réserves car, dans les mécanismes des certificats tout n'est pas résolu. On peut citer comme exemples une lacune technique et une faille de sécurité [DNAC].

La lacune technique réside dans la révocation des certificats. En effet, cette révocation est basée sur une liste qu'il faut télécharger régulièrement ; ce qui est contraignant et lourd. Les standards en cours d'élaboration, pour accéder à cette liste dynamiquement et automatiquement, ne sont pas encore implémentés par exemple dans Netscape ou Internet Explorer.

Quant à la faille de sécurité, elle est beaucoup plus importante, et s'articule autour du problème de confiance. En effet, tout ce dispositif nécessite des procédures strictes et sérieuses dans la gestion des certificats pour assurer les services qui sont affichés. S'il s'avère que ces procédures ne sont pas fiables, cela risquerait de conduire à la génération de faux certificats. Si ces incidents sont trop nombreux, plus personne ne ferait confiance aux certificats qui n'auront plus aucune valeur et seront voués à l'échec.

Les *IGC*s dans leur présentation actuelle, ne suffisent plus aujourd'hui à contrôler les accès aux applications de manière efficace. Une solution est donc de les coupler avec des Infrastructures de Gestion des Privilèges (*IGP*), avec des certificats d'attributs.

Dans le chapitre suivant nous présentons les certificats d'attributs ainsi qu'une vue globale des *IGP*s.

Chapitre 3

3 Les certificats d’attributs et leur utilité

Résumé

Comme cela a été expliqué au chapitre précédent, la troisième version de X.509 est plus flexible que les précédentes grâce au champ d’extensions qui permettent aussi d’imposer des restrictions (voir chapitre 2, paragraphe 2.4.3). Une solution [ITUT-X509-00] a été définie pour résoudre et simplifier ces restrictions. Elle consiste à partager un certificat X.509 en deux certificats : un certificat d’identité qui contient des informations sur l’identitédu sujet et un certificat d’attributs qui contient des informations sur les attributs.

En plus de l’IGC, la recommandation X.509 [ITUT-X509-00] définit aussi une Infrastructure de Gestion de Privilèges (IGP) qui est une norme d’autorisation forte [IUT2004], fondée sur des certificats d’attributs et des autorités d’attributs. L’IGP sert à vérifier les droits et les privilèges des utilisateurs.

Ce chapitre présente, une vision globale des IGPs. Les infrastructures IGP et IGC utilisent des concepts analogues, mais les premières concernent l’autorisation tandis que les secondes concernent l’authentification [IUT2004]. Ce chapitre illustre également l’intérêt et les limites des certificats d’attributs. L’illustration des limites des certificats d’attributs est nécessaire pour faire paraître la nécessité de proposer une nouvelle approche pour la génération des certificats d’attributs "simplifiés", qui répondrait plus simplement et efficacement aux besoins des applications.

3.1 Introduction

En théorie, il est possible, grâce aux champs "*extension*" du certificat X.509v3, d’intégrer des informations relatives au profil du porteur du certificat (fonction dans l’entreprise, rôle vis-à-vis de telle ou telle application, pouvoir en termes de passage de commande ou de transaction, etc.). En interprétant ces informations, les applications ont la possibilité d’accorder ou pas le droit d’accès aux données et aux transactions associées à cet utilisateur.

En pratique, il est pourtant peu recommandé d’utiliser ces méthodes pour gérer les habilitations applicatives. En effet, le profil applicatif d’un utilisateur est appelé à évoluer fréquemment (changement de fonction, accès à une nouvelle application, etc.), ce qui impliquerait :

- Soit de révoquer et recréer fréquemment des certificats pour chaque utilisateur et de les mettre à disposition sur un serveur de gestion des habilitations centralisé,
- Soit de diffuser à chaque utilisateur, en plus de son certificat d’identité, des certificats d’attributs pour chaque application ou pour chaque groupe d’utilisateurs (à détailler après).

Dans les deux cas, les opérations d’enregistrement seraient multipliées et la gestion des certificats deviendrait très contraignante pour les utilisateurs.

3.2 Du certificat d'identification au certificat d'attributs

3.2.1 Les groupes *PKIX* et *SPKI* de l'*IETF*

Des groupes de travail de l'*IETF* [IETF] tels que *PKIX* [RFC3647] et *SPKI* [RFC2692] considèrent dans leurs travaux la structure et le traitement de certificat. Créé en 1995, le groupe *PKIX* a pour but de développer un ensemble de normes définissant, spécifiquement pour l'Internet, une Infrastructure de Gestion des clefs (*IGC*) basée sur l'utilisation des certificats X.509 [HSC]. Le premier sujet de travail a été la définition d'un profil pour les certificats X.509, profil destiné aux différents protocoles applicatifs d'Internet (*SSL/TLS* [RFC2246], *S/MIME* [RFC3851], *IPSec* [RFC2401], etc.). Onze versions intermédiaires ont été nécessaires pour permettre au groupe de s'accorder sur un format acceptable qui fut édité sous la référence [RFC2459] en janvier 1999. En juillet 2001, une nouvelle version de cette norme a été éditée afin de clarifier certains points et d'en développer d'autres.

Une seconde norme de référence a été le RFC 2527 [RFC2527] de mars 1999. Cette norme définit un format type de politique de certification et de déclaration des pratiques de certification. C'est sur cette base qu'ont été rédigées la majorité des politique de certification disponibles publiquement [EYRAUT].

Dans le même temps, plusieurs protocoles de gestion des informations échangées dans l'Infrastructure de Gestion des Privilèges ont été développés comme, entre autres, *CMP* (*Certificate Management Protocol*) [RFC2510] et *CRMF* (*Certificate Request Message Format*) [RFC2511].

Le groupe *PKIX* a longuement débattu des processus de révocation et de contrôle des listes de certificats révoqués (*LCR*). Plusieurs protocoles ont été proposés pour mettre en œuvre ce service qui est essentiel à la bonne vérification de l'état des certificats. *CMP* supporte les requêtes de révocation et leurs réponses, et les messages de récupération de *LCR*. Le protocole *OCSP* (*Online Certificate Status Protocol*) [RFC2560] a été développé pour traiter les besoins de contrôle en ligne de l'état d'un certificat donné ce qui permet d'obtenir une information plus à jour qu'en passant par la consultation d'une *LCR*. Plus tard, le protocole *SCVP* (*Simple Certification Verification Protocol*) [MALDraft] a été produit pour permettre aux utilisateurs de se décharger de toutes les vérifications de certificats sur une autre entité (par procuration) [EYRAUT].

Depuis quelque temps, le groupe a travaillé sur les certificats d'attributs et sur les profils de certificats qualifiés [RFC3280] pour supporter les exigences de la directive européenne pour la reconnaissance légale de la signature électronique. Le groupe *PKIX* a intégré les certificats d'attributs dans le standard relatif au certificat numérique [RFC3280] pour créer le standard [RFC3281].

Contrairement au groupe *PKIX* qui se base sur la norme d'usage courant X.509 pour l'appliquer au monde Internet, le groupe *SPKI* [RFC2692] [RFC2693] vise à définir une infrastructure de gestion des clefs, et notamment un format de certificats, propre à l'*IETF* [HSC].

Le travail du groupe *SPKI* a commencé, en 1996, par la définition de l'ensemble des besoins [RFC2692] auxquels devra répondre l'*IGC*. En particulier, *SPKI* abandonne l'idée qu'un certificat permette de lier une clef à une identité pour considérer que le rôle d'un certificat soit plus général et attribue également des permissions au possesseur d'une clef. Un certificat *SPKI* contient donc un ensemble d'attributs et d'autorisations, éventuellement à caractère confidentiel. Un certificat *SPKI* relie une autorisation à une clef publique (voir figure 3-1), sans exiger nécessairement l'identité du détenteur de la clef privée correspondante. *SPKI* offre une simplicité par rapport au standard X.509 en utilisant un schéma d'encodage "*S-Expressions*" [SEXP] plus accessible et simple que la notation *ASN.1* utilisée par X.509. Les "*S-Expressions*" sont similaires au langage *XML* [RFC3470] et sont de haut niveau (compréhensibles), contrairement aux encodages *ASN.1* et *DER* utilisés par X.509. Les certifictas *SPKI* sont orientés vers l'autorisation et l'anonymat du propriétaire. Un certificat ou autre objet *SPKI* peut notamment comporter, suivant sa fonction, tout ou une partie des cinq champs suivants : émetteur (*Issuer*), sujet (*Subject*), permission de délégation (*Delegation permission*),

autorisation (*Authorization*), dates et/ou tests de validité (*Validity dates and/or tests*). Le champ sujet contient la clef publique et le champ autorisation les autorisations correspondantes (voir figure 3-1).

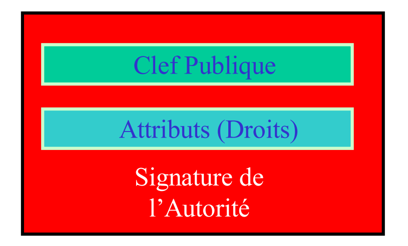


Figure 3-1 Certificat d'autorisation SPKI

SPKI a été proposé pour devenir une alternative au X.509 basé sur les *PKIX* [RFC3647]. Pour les petits systèmes, les propositions de *SPKI* offrent une vitesse de traitement élevée, contrairement aux Infrastructures de Gestion des Clefs qui supposent que le client a toutes les puissances de calcul et de stockage nécessaires pour effectuer les opérations exigées. Typiquement, les réponses authentifiées exigent du client de vérifier au moins une signature numérique sur la réponse d'une autorité de certification, ce qui consomme du temps, particulièrement si le décodage *DER* a été utilisé, comme avec les certificats X.509 et les systèmes associés.

Cependant, malgré sa grande souplesse d'utilisation et de mise en œuvre, le standard *SPKI*, utilisé pour des fins d'autorisation, ne s'est jamais imposé face à son concurrent X.509, largement déployé en natif dans les navigateurs. *SPKI* en soi ne se charge pas de la distribution des certificats. Il est cependant possible d'avoir recours pour cela à un système comme *DNSSEC* [RFC3845], au détriment de la confidentialité du contenu du certificat [HSC]. Son infrastructure décentralisée permet de mettre en œuvre de manière rapide une plate-forme de certification. Mais, les contraintes de supports des listes de contrôles d'accès, des noms *SDSI* [MITSDSI] et la non distribution des certificats ont limité le développement du standard *SPKI*.

3.2.2 Le groupe *SDSI* du *MIT*

SDSI [MITSDSI] est l'acronyme de *Simple Distributed Security Infrastructure*. C'est un modèle de *IGC* créé dans le laboratoire des sciences informatiques du *MIT* [LCS] en 1996. *SDSI* combine une *IGC* avec une méthode de définition de groupes et de publication de certificats. *SDSI* simplifie la terminologie pour définir des listes de contrôle d'accès et de politiques de sécurité. Il met en évidence la liaison entre les espaces de noms locaux (*Local Name Space*) plutôt que de favoriser la hiérarchie de l'espace globale de X.500 [ITUT-X500-01].

3.2.3 Unification de *SPKI* et *SDSI* dans l'*IETF*

La spécification du groupe *SPKI* de l'*IETF* et celle de *SDSI* du *MIT* se sont regroupées en 1997, en associant les avantages des deux spécifications. La spécification *SPKI*/*SDSI* utilise les noms locaux (identifiables dans un espace local) pour identifier et/ou autoriser un ou plusieurs utilisateurs de certificats. L'*ETSI* (*European Telecommunications Standards Institute*) [ETSI] a produit un certain nombre de textes (même s'ils restent très proches de ceux de l'*IETF*) qui sont disponibles sur [ETSI]. En mars 2000, un document de travail [PAADraft] qui définit une forme standard pour encoder les certificats *SPKI* en *XML* a été déposé à l'*IETF*. En novembre 2001, un autre document de travail proposait une grammaire *XML* pour décrire les certificats *SPKI* [ORRDraft].

Cette démarche a l'inconvénient de cibler la simplicité plutôt que les besoins réels des applications ou classes d'applications. Ce standard ne connaît pas aujourd'hui le succès attendu.

3.3 Les certificats d'attributs X.509

3.3.1 Introduction

Pour permettre l'ajout d'informations, telles que des attributs ou des privilèges, dans un certificat d'identité, la version trois de X.509 a introduit des options d'extension sous forme de blocks d'informations. Ces extensions permettent de spécifier des informations en fonction de l'usage que l'on souhaite donner au certificat. Malheureusement, l'ajout de ces extensions a induit des problèmes d'interopérabilité et de gestion de la révocation.

Une solution a été définie pour résoudre et simplifier le problème de révocation. La solution proposée par l'ITU-T (*International Telecommunication Union - Telecommunication Standardization Sector*) [ITUT] est de scinder un certificat X.509 en deux (voir figure 3-2) : un certificat d'identité qui consigne des informations sur l'identité et un certificat d'attributs [ITUT-X509-00] qui enregistre des informations sur les attributs (autorisation, contrôle d'accès). Cette solution simplifiera énormément le processus d'émission des certificats et pourra, dans certaines situations, éliminer le problème de la révocation. Les certificats d'attributs ayant une durée de vie très courte, leur révocation n'est pas obligatoire, ils expirent tout simplement.

Comme l'ont montré les résultats des études entreprises par le GIP-MDS [GIPMDS] et par le projet ICARE [ICARE] auquel nous avons participé, la séparation certificat d'identité / certificat d'attributs est nécessaire car :

- Au niveau logique, la durée de vie des attributs est différente de celle du certificat d'identité. En effet, si un directeur commercial d'entreprise perd sa position dans la hiérarchie interne, ses attributs sont invalidés et par conséquent son certificat; pourtant il a toujours droit à une signature électronique stricto sensu.

- Au niveau technique, les attributs sont garantis par diverses autorités qui n'ont pas nécessairement de relations avec le certificateur ayant émis le certificat : toute organisation a vocation à certifier l'appartenance et la position de ses membres en son sein.

- Enfin les attributs ne sont pas nécessairement demandés au même moment que la demande du certificat d'identité.

Le certificat d'identité X.509 dans sa forme la plus classique ne se rend pas compte de la qualité du professionnel. Aussi rencontre-t-on des services où seuls les professionnels préalablement identifiés comme tels peuvent y accéder.

Le certificat d'identité X.509 lie la clef publique à l'identité du porteur mais pas explicitement à ses à ses privilèges. Les certificats d'attributs ont pour objet de permettre de certifier les caractéristiques d'une personne, en particulier ses rôles ou droits d'accès, sans qu'elle ne doive modifier son certificat d'identité. Ils sont remis par une autorité dite «Autorité d'Attribution» (*AA*), qui, en général, est différente de l'émetteur du certificat d'identité.

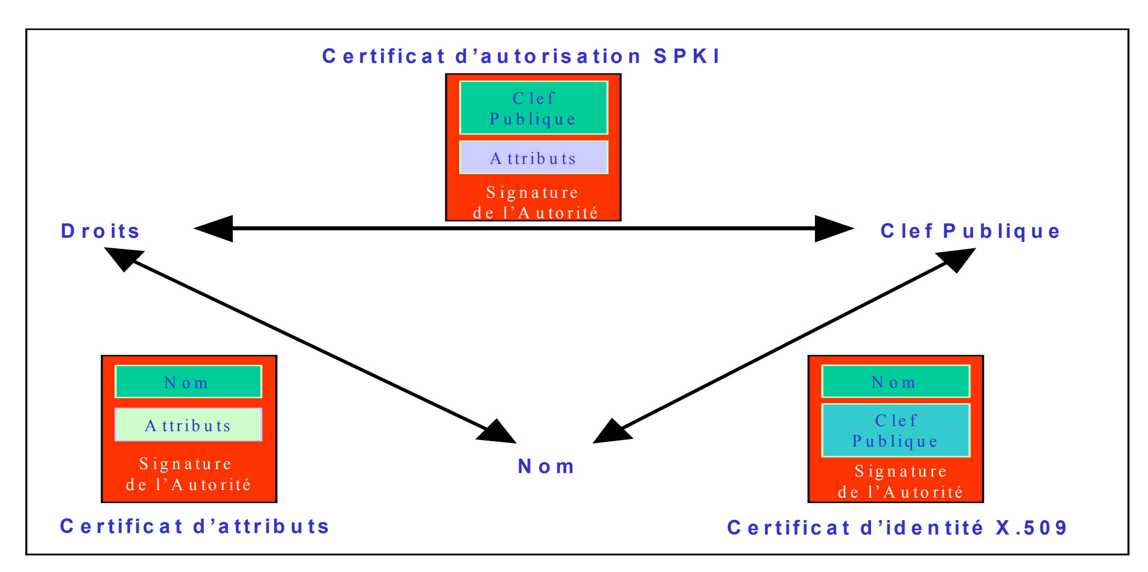


Figure 3-2 Intersection entre les certificats d'identité X.509, *SPKI* et Attributs

3.3.2 Définition

Le certificat d'attributs [ITUT-X509-00] constitue une réelle évolution. Il se présente comme un instrument technique permettant d'ajouter des fonctionnalités à la signature électronique classique [IALTA].

La norme X.509 donne la définition suivante d'un certificat d'attributs : «Un ensemble d'attributs d'un utilisateur associé à un ensemble d'autres informations, rendu non modifiable par la signature numérique utilisant la clef privée de l'autorité de certification qui l'a générée» [EYRAUT].

3.3.3 Services offerts par les certificats d'attributs

Un certificat d'attributs peut, à l'égard d'une personne, servir à établir notamment sa fonction, sa qualité, ses droits, ses pouvoirs ou privilèges au sein d'une personne morale, d'une association, d'une société, de l'État ou dans le cadre d'un emploi. À l'égard d'un document ou d'un autre objet, il peut servir à confirmer l'information permettant de l'identifier ou de le localiser ou de déterminer son usage ou le droit d'y avoir accès ou tout autre droit ou privilège afférent. L'accès au certificat d'attributs relatif à une personne doit être autorisé par celle-ci ou par une personne en autorité par rapport à elle.

L'utilisation du certificat d'attributs rend possible la délégation des droits, la signature avec un rôle et le contrôle de la multisignature électronique [IALTA]. Dans ce contexte, nous pouvons considérer trois grands services : l'habilitation/la délégation, la certification de rôles et la multisignature électronique contrôlée.

3.3.3.1 L'habilitation/la délégation

L'habilitation donne l'autorisation à une entité (généralement un subordonné) d'exercer un pouvoir à sa place. La délégation donne l'autorisation de transférer ce pouvoir à un tiers. Il faut remarquer qu'on peut habiliter tout ou une partie de ce pouvoir avec un attribut particulier. Dans le cas de l'habilitation de signature, le type de document à signer (feuille de congé, réservation de salle, etc.) peut aussi être pris en compte.

Pour pouvoir obtenir une telle habilitation, les certificats d'attributs peuvent être employés. Une entité A fournit un certificat d'attributs à une entité B, pour qu'elle puisse effectuer en son nom des actions pendant une durée déterminée. Par exemple, A permet à B de signer un document en son nom

avec un certificat. Ce certificat est alors joint à tout document signé par B à la place de A, il prouvera que B a bien le droit de signer. Le certificat d'attributs de B est composé de :

- ses droits,
- son identificateur,
- le temps de validité des droits,
- l'identifiant de A,
- la signature de A, la capacité de B à déléguer à un tiers.

B peut donc, dans certains cas et dans les mêmes conditions, habiliter à C et/ou D la signature de A, et ainsi de proche en proche pour créer une chaîne de délégation de signature dans laquelle le niveau de confiance ne se dégrade pas. L'identité des entités est alors garantie (sur demande) par l'émetteur du certificat d'attributs. Il vérifiera les certificats d'identité au moment de faire l'habilitation.

Au total, ce schéma se présente comme une solution dans laquelle le propriétaire de l'application n'a pas besoin de stocker les droits des personnes autorisées (ou uniquement ceux d'un responsable par entreprise), il a uniquement besoin de connaître le haut de la hiérarchie :

- soit parce que, comme expliqué plus haut, cette personne a des droits juridiques, par exemple, en tant que patron de l'entreprise, droits que l'application peut vérifier en s'adressant à un référentiel officiel,
- soit par enregistrement classique dans un fichier, mais uniquement pour une personne par entreprise.

Ensuite, tel qu'expliqué, la chaîne de signature de délégations de droits permet de vérifier les droits de l'utilisateur final. Cependant, cette solution nécessite la vérification de toute une chaîne de signatures.

3.3.3.2 La certification de rôles

Afin d'associer un pouvoir à une personne, en particulier le droit de signer, la sécurité emploie le concept de rôle. Une identité peut jouer un ou plusieurs rôles, comme elle peut ne jouer aucun. Dans ce contexte, il y a quatre scénarios possibles :

- Il existe plusieurs entités liées à un rôle.
- Il existe une seule entité liée à un rôle.
- Il existe plusieurs rôles liés à une entité.
- Il existe une entité sans rôle.

Le porteur du certificat d'attributs peut être une entité quelconque. Son identification peut être notamment un rôle. Ce rôle est la représentation des privilèges du signataire dans sa fonction. Il a la possibilité de garder l'anonymat dans certaines transactions. Ces rôles constituent une liaison indirecte entre les identités et leurs prérogatives. Ainsi, chacun peut choisir de déléguer (ou de ne pas déléguer) sa signature ou bien seulement une partie du pouvoir associé à cette signature.

Prenons pour exemple les rôles suivants de A :

- Direction de projet,
- Direction de projet, signature des congés du personnel,
- Direction de projet, embauche du personnel,

Dans le cas d'habilitation de la signature pour une demande de congés, A délègue son rôle "Direction de projet, signature des congés du personnel" à B. En revanche, A ne va déléguer à personne son rôle "Direction de projet, embauche du personnel", ainsi il est sûr que personne ne pourra embaucher du personnel sauf lui-même. Il y a aussi possibilité de déléguer sa signature de manière totale, c'est-à-dire de transmettre l'ensemble de son pouvoir avec le rôle "Direction de projet".

Avec les certificats d'attributs, les rôles peuvent être assignés de manière dynamique, par exemple si un employé change de fonction, il faut seulement révoquer l'ancien certificat et générer un certificat d'attributs qui lui assigne sa nouvelle fonction (ou rôle). Cela permet de garder le même certificat d'identité pour la signature de documents. L'utilisation du certificat d'attributs permet ainsi la certification de rôles assumés par les individus.

3.3.3.3 Multisignature électronique contrôlée

La multisignature électronique (appelée aussi signature de groupe) se base sur les mêmes principes que ceux de la signature électronique classique. Elle est nécessaire quand plusieurs entités doivent signer un document, par exemple un bon de commande, un contrat de travail, un projet du groupe, etc. La multisignature contrôlée est un service de multisignature utilisant le certificat d'attributs. Celui-ci permet d'étendre la multisignature d'un document en ajoutant des autorisations ou des contraintes particulières. L'application contrôle la séquence et les priorités des signatures.

Dans ce service, on attache un certificat d'attributs à un document. Ce certificat indique les entités (clefs publiques ou identificateur) qui peuvent signer le document et établit aussi l'ordre dans lequel les signataires doivent signer le document. Les informations essentielles à protéger par cette signature sont :

- Le contenu de la transaction.
- L'heure et la date de la signature (Horodatage - "*Time Stamping*").
- Les politiques de signature (qui, quand, comment doit-on signer le document*).*

Des informations additionnelles peuvent être indiquées pour faciliter le caractère légal de la signature :

- La référence des certificats qui valide la signature.
- Le type de transaction (afin d'appliquer les règles valables)
- Le lieu de la signature (afin de savoir quel droit appliquer pour la défense du signataire)

3.3.4 Structure des certificats d'attributs

La structure d'un certificat d'attributs (voir figure 3-3 et 3-4) est relativement similaire à celle d'un certificat d'identité, à la différence fondamentale que le certificat d'attributs ne contient pas de clef publique. Le certificat d'attributs est une structure de données signée par l'Autorité d'Attribution et contient la référence du certificat d'identité et les données décrivant les privilèges associés.

Parce qu'il ne contient pas d'informations explicites sur l'identité de la personne, un certificat d'attributs ne peut pas être utilisé pour authentifier le possesseur du certificat. C'est pourquoi le possesseur d'un certificat d'attributs doit aussi avoir un certificat à clef publique, auquel il est fait référence dans le certificat d'attributs.

Version	Numéro de version (version 2 pour la norme X.509, revue 2000).
Holder	Identité du porteur du certificat d'attributs.
baseCertificateID	Identifiant du porteur par référence à un certificat de clef publique (nom de l'*AC* et numéro de série du certificat d'identité du porteur auquel ce certificat d'attributs est associé).
entityName	Le nom du porteur.
objectDigestInfo	Une empreinte numérique d'informations liées au porteur.
issuer	Identité de l'autorité d'attribut (*AA*).
issuerName	Identifiant de l'*AA*.
baseCertificateID	Identifiant de l'AA par référence à un certificat de clef publique.
ObjectDigestInfo	Identifie l'AA par l'empreinte numérique d'informations propre à l'*AA*.
signature	Identifie l'algorithme utilisé pour signer le certificat d'attributs.
serialNumber	Numéro de série unique dans le contexte de l'*AA*.
attrCertValidityPeriod	Période durant laquelle le certificat est considéré comme valide.
notBeforeTime	Début de période de validité du certificat d'attributs.
notAfterTime	Fin de période de validité du certificat d'attributs.
attributes (séquence d'attributs)	Ce champ contient un ensemble d'attributs qui doivent être reconnus à l'intérieur du domaine qui concerne un porteur de certificat.
issuerUniqueID	Identifie de façon unique le fournisseur du certificat d'attributs, dans le cas où le nom de ce fournisseur ne serait pas suffisant. Ce champ est optionnel.
extensions	Permet d'ajouter de nouveaux champs à un certificat d'attributs. Ce champ est optionnel.

Figure 3-3 Structure d'un certificat d'attributs

Le formalisme *ASN.1* simplifié de certificat d'attributs est comme suit :

- *CertificatAttribut ::= SEQUENCE {*
- *Version* (version de la norme définissant le format du certificat).
- *Titulaire* (nom du titulaire sous une forme non ambiguë ; peut être soit le numéro de série et l'émetteur du certificat d'identité, soit la description d'un point de référence d'une personne morale [son nom de domaine Internet par exemple]).
- *Emetteur* (nom et référence unique de l'émetteur du certificat d'attributs).
- *AlgorithmeSignature* (référence des algorithmes utilisés pour signer le certificat d'attributs).
- *PeriodeValidité* (couples de dates «Pas avant, Pas après»).
- *Attributs SEQUENCE Atttribut* (liste des attributs alloués au *Titulaire* ; ce sont des informations, de nature variée, liées au type de l'application qui les utilise ; par exemple, le montant maximum d'un contrat que le titulaire peut signer).
- *IdentificateurUniqueEmetteur* (identificateur unique au cas où *Emetteur* pourrait être une information ambiguëe).
- *Extensions* (informations optionnelles liées à l'administration générale du système de signature et à l'application qui l'utilise).
- *SignatureCertificatAttribut* (valeur de la signature du certificat effectuée avec la clef privée de l'autorité d'attribution).

3.3.5 Comparaison entre un certificat d'attributs et un certificat d'identité

Les certificats d'identités et les certificats d'attributs sont tous les deux définis par la norme X.509. Bien que similaire en structure au certificat à clef publique, les informations fournies dans un certificat d'attributs tendent à être valides pour une période de temps limitée, éventuellement une heure ou moins. Elles ont également tendance à avoir un périmètre plus réduit ; par exemple elles peuvent ne concerner qu'un petit groupe comme une entité locale. En revanche les informations dans les

certificats à clefs publiques tendent à être valides pour des périodes de temps plus longues et à être utilisables de façon plus large.

Les certificats à clef publique prouvent l'identité de la personne mais ne définissent pas ce que la personne peut faire. Les certificats d'attributs ont été conçus pour répondre à cette problématique.

Parce qu'il ne contient pas d'informations explicites sur l'identité de la personne, un certificat d'attributs ne peut pas être utilisé pour authentifier le possesseur du certificat. C'est pourquoi le possesseur d'un certificat d'attributs doit avoir aussi un certificat d'identité, auquel il est fait référence dans le certificat d'attributs.

Les certificats d'attributs ne sont pas forcément émis par l'Autorité de Certification qui a généré le certificat d'identité. Différents certificats d'attributs pourront être produits par plusieurs *AA* qui n'ont aucun lien entre elles. Un individu possèdera alors plusieurs certificats d'attributs associés à son certificat d'identité.

Le titulaire d'un certificat d'identité peut disposer de plusieurs certificats d'attributs correspondant à des applications, autorités d'attribution et privilèges variés.

3.4 L'Infrastructure de Gestion des Privilèges

Plusieurs systèmes utilisent les certificats à clef publique (certificats d'identité) pour contrôler l'accès des utilisateurs, dès lors qu'il suffit de vérifier l'identité du propriétaire du certificat. Mais de plus en plus de systèmes exigent des règles d'accès qui ne sont pas présentes dans les certificats à clef publique, ni dans leurs extensions. C'est pour donner cette information que les certificats d'attributs ont été créés, ainsi qu'une infrastructure pour les administrer, l'Infrastructure de Gestion des Privilèges (*IGP*) ou *PMI* en anglais (*Privileges Management Infrastructure*).

Par définition, l'*IGP* est une infrastructure de gestion de privilèges, de matériel, de logiciel, de personnes, de politiques et de procédures, nécessaire pour créer, administrer, stocker, distribuer et révoquer les certificats d'attributs (voir figure 3-4) [RFC3281].

3.4.1 L'architecture et les services offerts par l'*IGP*

Les certificats d'attributs visent principalement à permettre la gestion des privilèges d'un individu. Cette approche est décrite dans la révision de 2000 de la norme X.509. L'infrastructure de gestion de privilèges est basée sur des autorités d'attributs (*AA*) qui ont pour rôle d'accorder des droits à des individus préalablement identifiés et authentifiés par leur certificat d'identité.

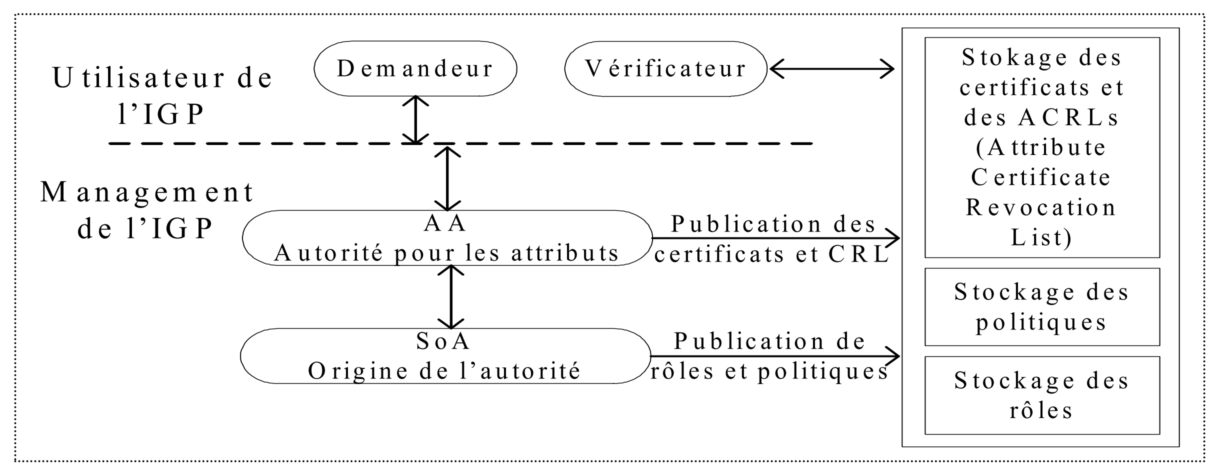


Figure 3-4 Architecture simplifiée d'un *IGP*

L'*IGP* permet de gérer les cas suivants :

- Le contrôle d'accès : les droits d'accès sont conférés à une personne par une *AA* qui gère les droits d'accès à un système donné. *L'AA* crée et signe un certificat d'attributs qui contient le droit d'accès accompagné d'une période de validité pour ce certificat. À l'échéance de la période de validité, le certificat d'attributs devient inutilisable.
- Les capacités de non répudiation : utilisées avec une politique de signature ou de non répudiation, les certificats d'attributs permettent de limiter les droits des utilisateurs.
- La délégation de droits : dans certains environnements, il est nécessaire de pouvoir déléguer des droits d'action à d'autres personnes, l'exemple le plus simple est la délégation de signature.
- La gestion de rôles : un certain nombre de droit d'accès ou d'actions peut être basé sur le rôle qui est joué dans l'entreprise (acheteur, comptable, responsable système, etc.). L'*AA* peut alors accorder un rôle à une personne au travers d'un certificat d'attributs.

3.4.2 Les composantes de l'*IGP*

Une *IGP* est composée des entités suivantes (voir la figure 3-4):

- L'Origine de l'autorité (*SoA*, "*Source of Authority*") : La Source d'Autorité *SoA* est l'entité vue par un vérificateur de privilèges comme étant l'entité ayant la responsabilité ultime pour l'attribution d'un ensemble de privilèges. Une *SoA* est elle-même une *AA* du moment qu'elle émet des certificats pour d'autres entités. Une *SoA* est analogue à une *CA root* du fait qu'un vérificateur de privilèges a confiance en les certificats signés par la *SoA*.
- L'Autorité pour les Attributs (*AA*) : C'est une entité qui peut générer un certificat d'attributs, et compte tenu de la délégation, si la SoA lui a donné cette autorisation, elle peut être un simple utilisateur.
- Le Demandeur : C'est l'entité qui émet la requête des attributs.
- Le Vérificateur : C'est l'entité qui vérifie les attributs du demandeur :
    1. vérifie l'origine du certificat (remonte jusqu'à la *SoA*),
    2. extrait les droits, en passant par la relation rôle -> privilège si nécessaire,
    3. les vérifie contre la politique d'accès,
    4. donne ou refuse le droit au demandeur,
    5. peut authentifier le demandeur.

Certains éléments sont très importants dans l'architecture :

1. **Le rôle** qui est une «collection» des privilèges permet d'ajouter un niveau d'abstraction dans les droits présentés par le demandeur.
2. **Les politiques d'accès (*Policy*)** : La *SoA*, en plus de son rôle d'émetteur d'attributs au demandeur, peut décrire des politiques avancées d'accès grâce à un langage abstrait. Ceci permet de développer une *API* pour le vérificateur indépendant des différentes ressources et façons d'y accéder. Le vérificateur accède à un module contenant les politiques et vérifie que les conditions décrites étaient valides pour permettre l'accès.

Certains éléments ne figurent pas directement dans l'architecture, ils peuvent être considérés comme optionnels :

1. Le serveur Web pour des interfaces http aux services,

2. Les bases de données pour stocker:
   a) les associations rôles-attributs,
   b) les certificats de rôles pour les employés,
   c) les politiques d'accès,
   d) les attributs définis.
3. Module de régénération automatique des certificats d'attributs.

3.4.3 Relation entre Autorité d'Attributs et Autorité de Certification

L'Autorité d'Attributs et l'Autorité de Certification sont logiquement (et dans plusieurs cas, physiquement) complètement indépendantes. La création et la maintenance de l'identité peuvent (et souvent doivent) être séparées de l'*IGP*. Ainsi, l'*IGC* entière, incluant les Autorités de Certification peut exister et être opérationnelle avant l'établissement de l'*IGP*.

Les spécifications de l'ITU-T couvrent plusieurs types d'environnements :

1. Dans plusieurs environnements, tous les privilèges seront affectés directement aux entités par une *AA* unique, à savoir la *SoA* ;
2. Dans d'autres environnements, les individus sont en possession de certificats leur affectant des rôles. Les privilèges associés aux rôles sont implicitement attribués à ces individus ;
3. Dans d'autres situations, un support de délégation doit être fourni. Dans une telle situation, la SoA affecte les privilèges à une entité pouvant agir elle-même comme une *AA* et par suite pouvant déléguer des privilèges ;
4. Parfois, la même entité physique peut agir comme *CA* et *AA*. Ce cas se présente quand les privilèges sont transportés par l'extension "*subjectDirectoryAttributes*" d'un certificat à clef publique. Dans d'autres environnements, des entités physiques séparées agiront comme *CA* et *AA*. Dans ce dernier cas, les privilèges sont affectés en utilisant les certificats d'attributs plutôt que les certificats à clef publique.

3.4.4 Principes de la gestion des attributs/des certificats

Les attributs peuvent être gérés soit au sein des entreprises, soit par un organisme extérieur reconnu comme tiers de confiance ; ce sera le cas pour l'appartenance à une profession réglementée où la gestion des attributs sera effectuée sous l'autorité de l'Ordre professionnel. Dans tous les cas, un responsable de distribution s'avère nécessaire. Cette personne prend en charge la création des attributs et la liaison possible avec un certificat de clef publique [SYNIAL].

3.4.4.1 Gestion au sein des entreprises :

Les entreprises peuvent gérer en interne leurs attributs dès lors qu'elles mettent en place une infrastructure adaptée.

L'intérêt est de délocaliser les politiques de validation des accès aux systèmes d'information et de s'affranchir de la gestion des utilisateurs.

Une telle gestion interne est cependant impraticable pour deux raisons :

1. elle nécessite une coûteuse gestion d'une architecture similaire à une *IGC*.
2. elle n'est pas a priori reconnue de l'extérieur et en particulier des clients étrangers lors de la production de documents à valeur légale. Cette reconnaissance peut être facilitée par une gestion externe des attributs.

3.4.4.2 Gestion extérieure

La validation d'une signature électronique par rapport à un contenu ne peut être complète qu'après examen et approbation d'un ensemble d'attributs déterminant les caractéristiques du signataire. Une gestion interne, bien que possible, est toutefois improbable. La gestion par un organisme externe de confiance, est ainsi privilégiée. Cet organisme délivre des attributs certifiés et dispose au sein des entreprises d'un acteur habilité à délivrer des attributs pour le compte de son entreprise appelé mandataire. Ces attributs sont alors certifiés par l'organisme extérieur dès lors que les attributs du mandataire sont valides. Le conteneur d'attributs (certificat d'attributs ou autre) devra alors référencer de manière non ambiguëe le mandataire pour des raisons de répudiation.

Bien que le passage par un tiers de confiance facilite la reconnaissance externe des attributs, en particulier vis-à-vis des clients et des organismes d'état, les problèmes suivants peuvent être soulevés:

1. La non-divulgation d'informations sensibles à l'extérieur. Les attributs doivent être protégés afin que le vérificateur ne puisse prendre connaissance que des attributs dont il a besoin et non de l'ensemble des attributs du signataire.
2. La gestion d'attributs destinés uniquement à un usage interne. Une caractéristique des attributs (telle qu'une extension critique) doit donc spécifier leur portée.

Une solution envisageable au premier point consiste à autoriser tout détenteur d'attributs à auto-générer un sous-ensemble d'attributs.

3.4.5 Modèles relatifs aux certificats d'attributs

3.4.5.1 Le modèle général

Le modèle général de gestion de privilèges est composé de trois entités : l'objet, le privilège autorisé et le vérificateur de privilèges.

- L'objet peut être une ressource protégée, par exemple dans une application à contrôle d'accès. Ce type d'objet peut avoir des méthodes pouvant être invoquées (par exemple, l'objet peut être un *firewall* qui a une méthode "*Allow entry*", ou l'objet peut être un fichier système qui a les méthodes *read*, *write* et *execute*). Un autre type d'objet peut être un objet qui a été signé dans une application de non-répudiation.
- Le privilège autorisé est l'entité qui détient un privilège particulier et le présente dans un contexte particulier d'utilisation.
- Le vérificateur de privilèges est l'entité qui détermine si les privilèges présentés sont suffisants dans le contexte d'utilisation.

3.4.5.2 Le modèle de délégation

Pour certaines implémentations, il peut être nécessaire de déléguer des privilèges. La Recommandation UIT-T X.509 définit un modèle de délégation d'architecture *IGP* à quatre composants: le vérificateur de privilège, la source d'autorité (*SoA*), d'autres autorités d'attribut (*AA*) et le déclarant de privilège (voir figure 3-5).

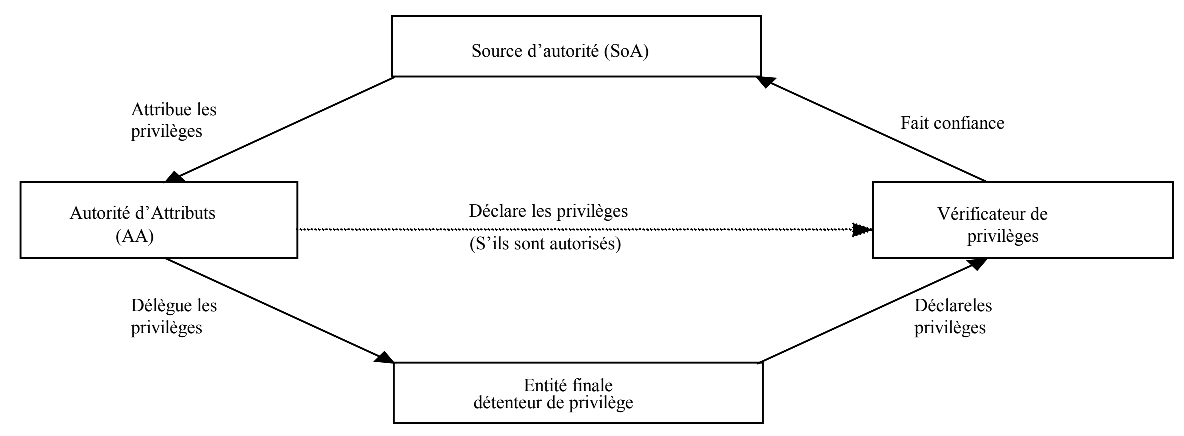


Figure 3-5 Le modèle de délégation

Dans certains environnements, la *SoA* qui est l'émettrice initiale des certificats affectant des privilèges à leurs détenteurs peut autoriser le détenteur de privilèges à agir comme une *AA* et par la suite déléguer ces privilèges à d'autres entités en émettant des certificats contenant les mêmes privilèges (ou un sous ensemble de ces privilèges).

La *SoA* peut imposer des contraintes sur la délégation (exemple : limiter la longueur de chemin de validation, limiter l'espace de noms dans lequel la délégation peut être faite). Chacune des *AAs* intermédiaires peut propager la délégation des privilèges.

Une restriction universelle sur la délégation est qu'aucune *AA* ne peut déléguer plus de privilèges qu'elle n'en possède. Une *AA* peut aussi restreindre les capacités de délégation des *AAs* situées à son aval.

Le vérificateur de privilèges a confiance en la *SoA* comme étant l'autorité pour un ensemble de privilèges de la ressource. Si le certificat contenant le privilège n'est pas issu de la *SoA*, alors le vérificateur de privilèges devra localiser un chemin de certification depuis le certificat contenant le privilège jusqu'à la *SoA*. La validation de ce chemin de délégation inclut la vérification que chaque *AA* a suffisamment de privilèges et a été dûment autorisé pour déléguer ces privilèges.

Un chemin de délégation sera formé exclusivement de certificats d'attributs ou de certificats à clef publique. Un délégateur qui a obtenu son privilège dans un certificat d'attributs peut déléguer ce privilège, s'il est autorisé à le faire, uniquement par le biais d'un certificat d'attributs. De façon similaire, un délégateur qui a obtenu son privilège dans un certificat à clef publique peut déléguer ce privilège, s'il est autorisé à le faire, uniquement par le biais d'un certificat à clef publique.

Le modèle de délégation décrit ci-dessus est celui défini dans le standard X.509. Le groupe de travail *PKIX* dans [RFC3281] ne recommande pas l'utilisation de chaînes de *ACs*. En effet, jugeant que l'administration et le traitement de chaînes de *ACs* sont complexes et que l'utilisation des ACs dans Internet est limitée, les spécifications du groupe de travail *PKIX* traitent le simple cas où une autorité émet toutes les *ACs* pour un ensemble particulier d'attributs. Cependant, cette simplification n'exclut pas le cas de l'utilisation de différentes autorités, chaque autorité ayant la responsabilité d'un ensemble d'attributs. Par exemple, l'appartenance à un groupe pourrait être incluse dans une *AC* issue d'une autorité, et l'attribution d'un rôle à une entité pourrait être incluse dans une *AC* issue d'une autre autorité.

3.4.5.3 Le modèle rôle

Les rôles fournissent un moyen pour attribuer indirectement des privilèges à des individus. En effet, des certificats d'attribution de rôles sont émis pour les individus, ils attribuent des rôles via l'attribut rôle y figurant.

Des privilèges spécifiques sont attribués aux rôles à travers des certificats de spécification de rôle plutôt qu'à travers les certificats d'attributs. Cette indirecte attribution permet, par exemple, de mettre à jour les privilèges attribués à un rôle, sans mettre à jour les certificats attribuant des rôles aux individus. Les certificats d'attribution de rôles peuvent être des certificats d'attributs ou des certificats à clef publique. Les certificats de spécification de rôle peuvent être des certificats d'attributs mais non des certificats à clef publique. Si les certificats de spécification de rôle ne sont pas utilisés, alors l'attribution des privilèges à un rôle peut être faite autrement (par exemple, en configurant manuellement le vérificateur de privilèges).

Les cas suivants sont possibles :

1. Une *AA* peut définir un nombre quelconque de rôles ;
2. Le rôle lui-même et les membres d'un rôle peuvent être définis et administrés séparément par différentes *AAs* ;
3. L'adhésion à un rôle, comme les autres privilèges, peut être déléguée ;

Si le certificat d'attribution de rôles est un certificat d'attributs, l'attribut rôle est contenu dans le champs *"attributes"* dans le certificat d'attributs. Si le certificat d'attribution de rôle est un certificat à clef publique, l'attribut rôle est contenu dans l'extension "*subjectDirectoryAttributes*". Dans le dernier cas, tous les privilèges additionnels contenus dans le certificat à clef publique sont des privilèges directement attribués au sujet du certificat et non des privilèges attribués au rôle.

Un privilège autoriser peut présenter un certificat de spécification de rôle à un vérificateur de privilèges pour prouver qu'il a un rôle particulier (par exemple *"Manager"*). Le vérificateur de privilèges peut connaître a priori, ou peut découvrir par différents moyens, les privilèges associés à ce rôle pour prendre des décisions sur l'autorisation. Un certificat de spécification de rôle peut être utilisé pour cet objectif.

3.5 Intérêt et limites du certificat d'attributs

Sur un plan purement technique, un certificat d'attributs est utilisable pour gérer des habilitations dans des structures complexes nécessitant la gestion des délégations. Le projet ICARE [ICARE] prouve que cela peut fonctionner techniquement. En mettant en place une gestion des certificats d'attributs adaptée (et relativement complexe), il est par exemple possible de s'assurer que la personne qui signe un document électronique a bien l'attribution ou la délégation pour le faire. Et il semble que cette gestion des délégations est un des avantages du certificat d'attributs par rapport à d'autres moyens utilisés pour procéder aux habilitations (annuaires…). Il convient toutefois de noter que l'utilisation de certificats d'attributs dans ce contexte, pour préserver un certain niveau de sécurité, nécessite la mise en place d'une infrastructure de gestion de certificats d'attributs, qui engendre un certain poids.

Les besoins en terme de signature électronique étant ponctuels, on peut envisager une infrastructure qui génère "à la volée" des certificats d'attributs d'une durée d'une journée de validité. Ces certificats seraient générés à partir d'un annuaire et archivés par l'ordre dans le cas des professions réglementées.

Un certificat d'attributs ne contient que des informations relatives aux attributs d'un individu. Il est donc généralement associé à un certificat d'identité. Le certificat d'identité est rattaché à un individu, le certificat d'attributs fait le lien avec l'entreprise. Un des avantages est alors qu'il n'est pas nécessaire de re-générer le certificat d'identité à chaque changement de société ou d'attribut, un nouveau certificat d'attributs étant généré dans ce cas.

Mais en quoi la génération d'un certificat d'attributs est-elle moins contraignante que celle d'un certificat d'identité ?

Pour que les partenaires puissent avoir confiance en un certificat d'attributs, il doit être généré avec un certain nombre de garanties, du même type que celles appliquées à un certificat d'identité. Il n'y a donc pas réellement de gain en matière de temps ou de simplification au moment de la génération. Toutefois, l'infrastructure de génération d'un certificat d'attributs pouvant être en partie indépendante de celle d'un certificat d'identité, cette séparation des pouvoirs peut être un argument en faveur de la sécurité dans des entreprises.

Par ailleurs, un certificat d'attributs est généré pour une période courte, ce qui permet de ne pas gérer les révocations. Un certificat d'attributs offre donc un gain en matière de gestion, mais cette caractéristique peut limiter ses utilisations possibles. Il n'est pas en effet forcément intéressant de re-générer sans cesse des certificats d'attributs pour exprimer un attribut pérenne pour un individu. La génération de certificats d'attributs ayant une durée de vie plus longue n'alourdirait-elle pas considérablement le processus ? De même, pour la gestion des révocations de ces certificats.

De plus, un certificat d'attributs étant associé à un certificat d'identité, il est parfois nécessaire, sur le plan de la sécurité, de contrôler ce certificat d'identité avant d'utiliser le certificat d'attributs correspondant. Cette démarche est devenue, dans la plupart du temps, une précaution technique que personne n'applique même lorsqu'elle n'est pas justifiée par des aspects juridiques. Sur un plan légal, le contrôle de la qualité (attribut) d'un individu est en effet souvent suffisant dans la vie «réelle» (non dématérialisée), alors que l'identité d'un individu est généralement requise dans les procédures de dématérialisation. Se pose alors le problème du droit d'un individu à l'anonymat, qui existe aussi dans le monde dématérialisé.

A un autre niveau, le concept même de séparation du certificat d'attributs et du certificat d'identité offre la possibilité de générer un certificat d'identité «unique» par individu, une multitude de certificats d'attributs étant par ailleurs générée par domaine ou activité.

Tel qu'il est défini, le certificat d'attributs semble donc offrir d'importantes possibilités pour un besoin toutefois assez ciblé, celui d'utilisations ponctuelles (voire uniques) par un individu couvrant de très nombreux domaines. Tant qu'il a une durée de vie si courte, il semble à première vue assez peu adapté à une gestion pérenne d'habilitations dans une entreprise. En rallongeant sa durée de vie, l'infrastructure à mettre en place serait nettement plus lourde, et nécessiterait une réelle étude du contexte, au cas par cas. Dans une entreprise, l'utilisation de certificats d'attributs semble à l'heure actuelle ne pouvoir répondre qu'à des besoins très ponctuels, ou bien nécessiter une infrastructure d'une certaine envergure.

Une courte durée de vie semble rendre le certificat d'attributs particulièrement bien adapté aux besoins des professions libérales. Toutefois, dans certains cas, certains contrôles (comme le contrôle du certificat d'identité) qui ne sont pas toujours justifiés juridiquement, alourdissent cette utilisation sans apporter de réelle valeur ajoutée.

Bien que les certificats d'attributs aient résolu et simplifié le problème de révocation, ils possèdent toujours les problèmes suivants [SSGRR03]:

1. Problème d'interopérabilité entre les applications : Causé par les extensions de certificats d'attributs qui peuvent être qualifiées de critiques ou non.
2. Problème de satisfaction des besoins de l'application et de l'utilisateur : A ce jour, les champs du certificat d'attributs spécifiés en *ASN.1* dans le standard sont fixés et ne correspondent pas toujours aux besoins de l'utilisateur et/ou de l'application.

Face à ces problématiques apparaît la nécessité de trouver une nouvelle approche pour la génération des certificats d'attributs.

Outre l'utilisation de certificats d'attributs «classiques», il pourrait être pertinent de définir un modèle de certificat d'attributs «simplifié», qui répondrait plus simplement et efficacement aux contraintes de ce type de profession.

3.6 Conclusion

Les certificats d'identité sont les pré-requis pour l'authentification de l'entité. Cependant, le certificat d'identité et la fonction d'authentification ne permettent pas de déterminer les droits de l'entité sur des ressources protégées.

Dans ce chapitre, nous avons présenté différents types de certificat. Les deux types de certificats les plus répandus sont le certificat d'attributs X.509 et le certificat *SPKI*. Chacun d'entre eux offre des services ressemblants mais avec des mécanismes et des formats différents.

Les certificats *SPKI* sont encodés en *S-expressions* et sont orientés vers l'autorisation et l'anonymat du propriétaire. Leur infrastructure décentralisée permet de mettre en œuvre de manière rapide une plate-forme de certification. Néanmoins, les contraintes de supports des listes de contrôles d'accès (*ACL*), des noms *SDSI* et la non distribution des certificats ont limitées son développement.

Nous avons exposé les certificats d'attributs X.509 (services offerts, usage, structure, ..). Un certificat d'attributs contient un ensemble d'attributs qui donnent des informations sur les privilèges du possesseur du certificat. Il convient toutefois de noter que l'utilisation de certificats d'attributs nécessite la mise en place d'une Infrastructure de Gestions des Privilèges (*IGP*) qui engendre une certaine lourdeur.

Nous avons présenté une vue globale de l'*IGP*. Cette infrastructure est centralisée et orientée sur l'autorisation.

Finalement nous avons illustré l'intérêt et les limites des certificats d'attributs. Les certificats d'attributs ont été créés pour résoudre les problématiques des certificats d'identités.

Les certificats d'attributs possèdent quelques points faibles. Une part de ces faiblesses est attribuable à l'encodage des certificats d'attributs X.509 en format *ASN.1* et à la difficile intégration de nouveaux attributs. A ce jour, les champs du certificat d'attributs spécifiés en *ASN.1* dans le standard [ITUT-X509-00] sont fixés et ne correspondent pas toujours aux besoins de l'utilisateur et/ou de l'application.

Une autre part de cette faiblesse est attribuable au problème d'interopérabilité entre les applications causé par les extensions de certificats d'attributs qui peuvent être qualifiées de critiques ou non.

Il serait donc pertinent de définir un modèle de certificat d'attributs simplifié, qui répondrait plus simplement et efficacement aux besoins de l'utilisateur et/ou de l'application.

Dans le chapitre suivant on propose et implémente une Extension de l'Infrastructure de Gestion de Privilège (*E-IGP*). L'*E-IGP* propose une nouvelle approche de certification qui génère automatiquement des certificats d'attributs simplifiés en satisfesant à la fois les besoins spécifiques de l'application et de l'utilisateur.

Chapitre 4

4 E-IGP : Extension de l'Infrastructure de Gestion des Privilèges

Résumé

Les Infrastructures de Gestion des Clefs (IGC) sont essentielles pour fournir les services de sécurité dans des réseaux ouverts comme l'Internet. Les services d'authentification fournis par les IGCs ne satisfont pas les besoins de beaucoup d'applications de commerce électronique. Ces applications nécessitent l'emploi des services d'autorisation supplémentaires afin de prouver ce que les utilisateurs ont le droit de faire. Pour contrôler l'accès à une ressource, par exemple, "l'authentification" et "l'autorisation" sont nécessaires.

En plus de l'Infrastructures de Gestion des Clefs, le standard X.509 définit une Infrastructure de Gestion de Privilège (IGP), qui est une norme d'autorisation forte, fondée sur des certificats d'attributs et des autorités d'attributs. Ainsi, afin d'avoir une infrastructure complète qui assure les deux services d'authentification et d'autorisation, la solution serait de coupler les deux infrastructures existantes, à savoir les IGC et les IGP.

Comme cela a été présenté dans le chapitre précédent, malgré les avantages des certificats d'attributs générés par une IGP, ceux-ci ne correspondent pas toujours aux besoins de l'utilisateur et/ou de l'application.

Ce chapitre propose et implémente une Extension de l'Infrastructure de Gestion de Privilège, appelée (E-IGP). Cette extension génère automatiquement des certificats d'attributs en XML. Ces certificats répondent à la fois aux besoins spécifiques des applications et des utilisateurs.

Notre approche consiste à créer des certificats d'attributs en XML spécifiques pour chaque application. Ces certificats contiennent uniquement l'information nécessaire pour l'exécution de cette application. Ainsi, pour chaque application, on définit une grammaire pour préciser tous les paramètres dont l'application a besoin. La génération de ces certificats d'attributs respectera la grammaire associée à l'application. Pour satisfaire les besoins de l'utilisateur, l'E-IGP permet à ce dernier de personnaliser sa demande de certificats d'attributs.

Les résultats de cette partie ont été sujets de plusieurs publications, notamment [SETIT04], [SAR03], [SETIT03], [SSGRR03], [SAR02].

4.1 Introduction

Les services d'authentification fournis par les Infrastructures de Gestion des Clefs (*IGC*s) ne satisfont pas les besoins de beaucoup d'applications. Ces applications nécessitent l'emploi des services d'autorisation supplémentaires afin de prouver ce que les utilisateurs ont le droit de faire. Pour contrôler l'accès à une ressource "l'authentification" et "l'autorisation" sont nécessaires.

En l'absence d'une solution complète de gestion des privilèges et d'autorisation, on se trouve rapidement face à un véritable problème administratif. Pour le commerce comme pour l'éducation, il y a des besoins diversifiés, par exemple, de procuration ou d'autorisation, d'où le besoin d'avoir des certificats de types "*Open*". A ce jour, les champs du certificat d'attributs spécifiés dans le standard sont fixés et ne correspondent pas toujours aux besoins de l'utilisateur et/ou de l'application. Ainsi, il est nécessaire de trouver une méthode pour remplir à la fois les vrais besoins de l'application (en précisant les paramètres dont l'application a besoin) et de l'utilisateur (en précisant la valeur des paramètres de l'application, les délégations, les rôles, la validité du certificat, etc).

Ce chapitre propose une infrastructure de génération automatique des certificats d'attributs en *XML* dans le but de satisfaire les besoins spécifiques de l'application et de l'utilisateur.

Dans ce chapitre, nous présentons une Extension de l'Infrastructure des Gestions des Privilèges (*E-IGP*) [SETIT04] qui répond à cette nécessité. Nous exposons une implémentation de cette extension, basée sur les certificats d'attributs spécifiés en *XML* et qui satisfont les vrais besoins de l'application et de l'utilisateur en même temps [SSGRR03]. L'extension proposée permet aux propriétaires et aux concepteurs de portails de marchés électroniques et d'entreprises d'utiliser une infrastructure de gestion des privilèges sécurisés et fournit aux clients des services d'accès sûrs et personnalisés.

4.2 L'extension proposée

Par définition, l'*E-IGP* [SAR03] est une extension de l'Infrastructure de Gestion des Privilèges qui crée, administre, stocke, distribue des certificats d'attributs simplifiés en format *XML* et administre les services associés (contrôle d'accès, délégation des rôles, etc.) à ces certificats d'attributs.

Les certificats créés par l'*E-IGP* satisfont exactement les besoins de l'application et de l'utilisateur en même temps. Notre approche consiste à créer des certificats d'attributs en *XML* (voir figure 4-1) spécifiques pour chaque application contenant uniquement l'information nécessaire pour la réalisation de cette application.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CertificatAttributs SYSTEM "Application1.dtd">
<CertificatAttributs Version="1">
  <Détenteur>....</Détenteur>
  <NomDeApplication>Application1</NomDeApplication>
  <BandePassante>2 Mbits/s</BandePassante>
  <ConnexionInternet>
            <ConIntType>Permanente</Type>
  </ConnexionInternet>
  <ConnexionAuReseau>
            <NomDeReseau>R</NomDeReseau>
            <ConResType>Permanente</Type>
  </ConnexionAuReseau>
   <CompteAccesDialUp>illimites</CompteAccesDialUp>
   ……………………..
   <AssignationsDeBP>
            <PossibiliteDeGestion>oui</PossibiliteDeGestion>
            <Temps>reel</Temps>
</AssignationsDeBP>
  <Validite>
   <DateDebut>2001.12.12.12.12</DateDebut>
   <DateFin>2002.12.12.12.12</DateFin>
  </Validite>
  <NumeroDeSerie>1012313281</NumeroDeSerie>
</CertificatAttributs>
```

Figure 4-1 Exemple d'un certificat d'attributs en *XML*

Pour chaque application, on définit une grammaire (voir figure 4-2) pour préciser tous les paramètres dont l'application a besoin. La génération de ces certificats d'attributs respectera la grammaire associée à l'application. L'idée principale est de stocker, dans la base de données des grammaires de l'*E-IGP*, des grammaires associées aux applications, c'est-à-dire des fichiers contenant un certain nombre de paramètres correspondant aux données qui seront insérées dans le certificat d'attributs final. Ainsi, si une nouvelle application a été ajoutée, c'est-à-dire le besoin d'un nouveau type de certificat d'attributs émane, il suffit que l'administrateur de l'*E-IGP* crée la grammaire correspondant à cette application et le sauvegarder dans la base de données des grammaires de l'*E-IGP*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- C'est la grammaire d'exemple de certificat d'attribut en XML-->
<!ELEMENT CertificatAttributs (NomDeApplication, BandePassante,
 ConnexionInternet, ConnexionAuReseau, EspaceDisque, ………
……..ReglesDeFiltrageDuPareFeu, AssignationsDeBP,
LienAvecSujet, Validite, NumeroDeSerie, UriApplication)>
<!ATTLIST CertificatAttributs
              Version CDATA #FIXED "1">
< !ELEMENT NomDeApplication (#PCDATA)>
<!ELEMENT BandePassante (#PCDATA)>
<!ELEMENT ConnexionInternet (ConIntType)>
<!ELEMENT ConIntType (#PCDATA)>
<!ELEMENT ConnexionAuReseau (NomDeReseau, ConResType)>
<!ELEMENT NomDeReseau (#PCDATA)>
<!ELEMENT ConResType (#PCDATA)>
<!ELEMENT EspaceDisque (Volume)>
<!ELEMENT Volume (#PCDATA)>
………………….
<!ELEMENT AssignationsDeBP (PossDeGestion, TempsAss)>
…………………..
<!ELEMENT LienAvecSujet (#PCDATA)>
<!ELEMENT Validite (DateDebut, DateFin)>
<!ELEMENT DateDebut (#PCDATA)>
<!ELEMENT DateFin (#PCDATA)>
<!ELEMENT NumeroDeSerie (#PCDATA)>
<!ELEMENT UriApplication (#PCDATA)>
```

Figure 4-2 Exemple de grammaire

Pour satisfaire les besoins de l'utilisateur, on permet à ce dernier de personnaliser sa demande de certificat d'attributs. C'est l'utilisateur qui précise les valeurs des paramètres de l'application, la date de validité de son certificat d'attributs, les rôles qu'il souhaite avoir ou les délégations (si elles existent) qu'il souhaite fournir à quelqu'un suivant ces besoins, etc.

4.3 Architecture de l'*E-IGP*

La mise en œuvre d'*E-IGP*, suppose l'existence d'une Infrastructure de Gestion des Clefs, auquelle l'*E-IGP* sera rattaché. L'IGC que nous avons utilisées est celle de Baltimore [BAL]. En effet, la gestion des autorisations ou des privilèges n'a de sens que si une phase d'authentification est préalablement effectuée.

L'*E-IGP* est composée de deux modules (voir figure 4-3). Le premier module "*GCAX* : Génération de Certificat d'Attributs en *XML* " permet la génération des certificats d'attributs simplifiés en *XML* tandis que le second "*V&CAA* : Vérification et de Contrôle d'Accès aux Applications" permet la vérification de ces certificats d'attributs ainsi que le contrôle d'accès aux applications de manière sécurisée. Ces deux modules (voir figure 4-4) sont des serveurs *Web* sécurisés indépendants reliés à une base de données qui contient les grammaires associées aux applications.

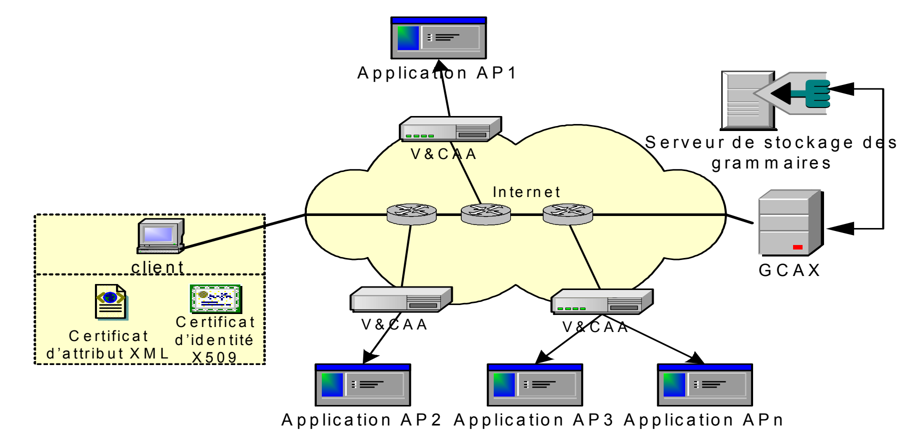


Figure 4-3 Les modules de l'architecture *E-IGP*

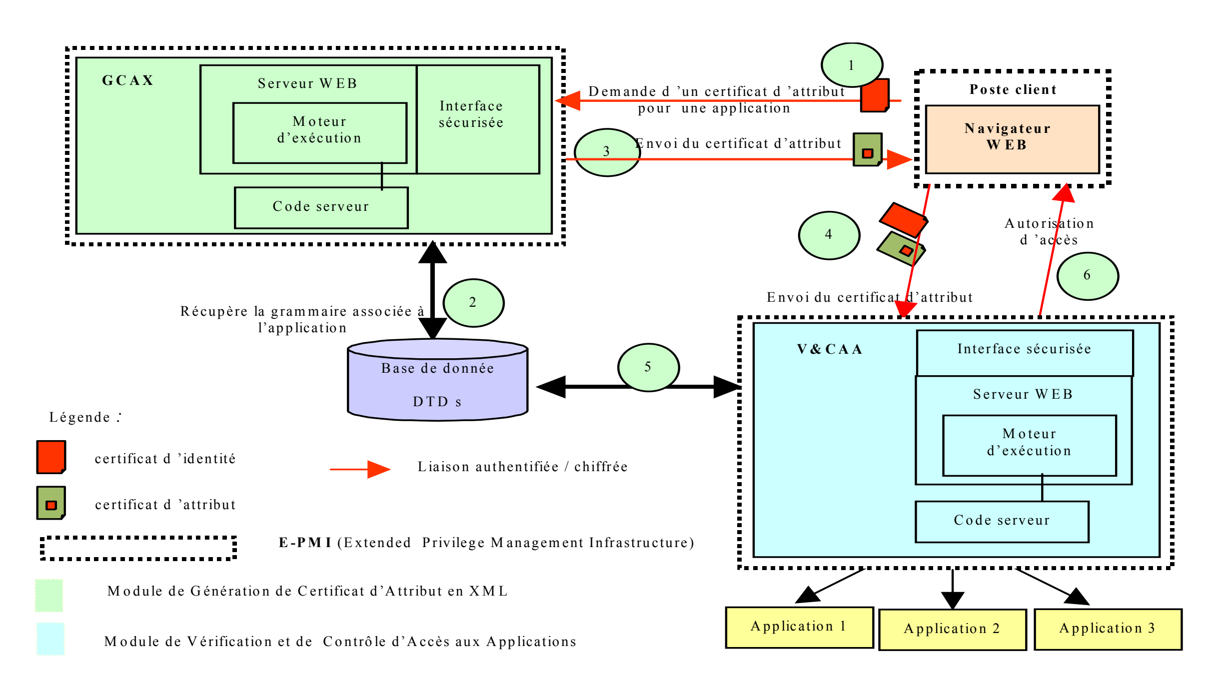


Figure 4-4 L'achitecture *E-IGP* proposée

4.3.1 Principe de fonctionnement

Un utilisateur qui désire accéder à l'une de nos applications doit avoir un certificat d'identité X509 et un certificat d'attributs généré par le *GCAX*. Ces deux types de certificat sont complémentaires. En effet, le possesseur d'un certificat d'attributs doit aussi avoir un certificat d'identité pour s'authentifier. Dans les sections suivantes, nous décrivons le fonctionnement des deux modules de l'*E-IGP*.

4.3.1.1 GCAX : Module de Génération de Certificat d'Attributs en XML

GCAX [SETIT04] est une entité qui a pour but d'accorder des droits aux utilisateurs pour des applications données. Le *GCAX* est l'interface qui prend en charge la génération de certificats d'attributs en *XML* suivant une politique donnée. *GCAX* est une interface interactive utilisant les technologies *JSP* [JAV] et *Servlet* [JAV].

Dans cette partie, nous décrivons les étapes que l'utilisateur devra suivre pour avoir un certificat d'attributs pour accéder à une application donnée (voir figure 4-5).

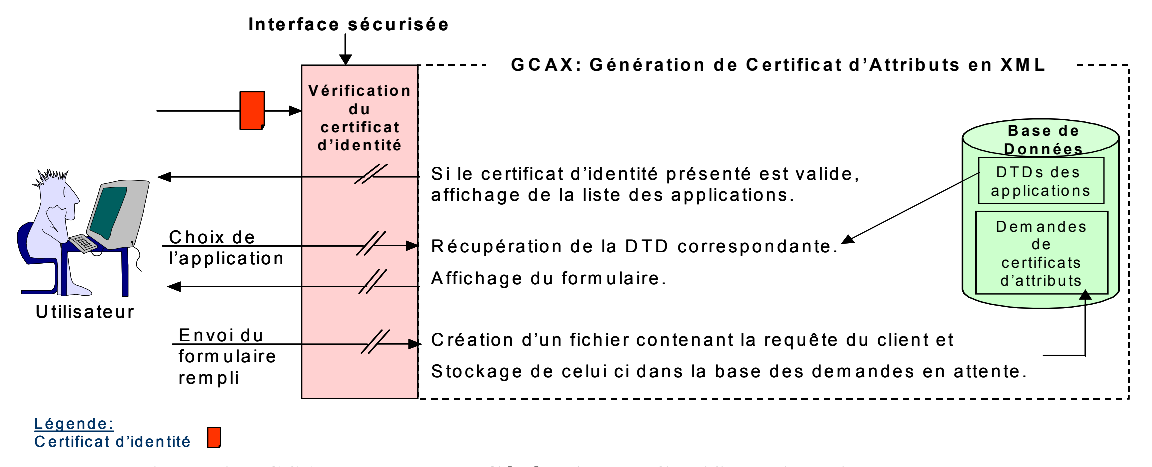


Figure 4-5 *GCAX* : Module de Génération de Certificat d'Attributs en *XML*

Pour que l’utilisateur puisse accéder au *GCAX*, il se connecte avec son navigateur *Web* via le protocole *HTTPS* (*HTTP* [RFC2616] sécurisé par le protocole *SSL* [RFC2246]). De ce fait, il utilise le protocole *SSL* pour ouvrir une connexion sécurisée avec le serveur. A travers *SSL*, l’utilisateur envoie son certificat d’identité X509 et s’authentifie à notre serveur *SSL*.

Une fois l’identité assurée et authentifiée, le *GCAX* visualise à l’utilisateur, à travers une page *Web*, les différents types d’applications qu’il possède. L’ensemble des applications pour lesquelles il est possible de demander des privilèges, ainsi que les privilèges associés, est référencé dans une base de données. L’utilisateur à travers cette page *Web*, choisit l’application qu’il désire accéder. Une fois le choix fait, *GCAX* récupère de sa base de données la grammaire associée à cette application et présente à l’utilisateur une interface qui visualise les différents champs ou options de cette application. C’est l’utilisateur qui précise les valeurs des paramètres de l’application, la date de validité de son certificat d’attributs, les rôles qu’il souhaite avoir ou les délégations (si elles existent) qu’il souhaite fournir à quelqu’un suivant ces besoins, etc.

Une fois que l’utilisateur spécifie l’application, pour laquelle il souhaite recevoir des privilèges particuliers et personnalise sa demande de certificat d’attibuts (voir figure 4-6), cette demande (ou formulaire de demande) est alors stockée dans la base de données des demandes en attente de *GCAX* qui parviennent automatiquement à l’administrateur. Ce dernier décidera sur l’accord ou le refus d’attribution d’un certificat d’attributs à l’utilisateur suivant sa politique appliquée. Une fois une requête est validée (accord d’attribution), l’administrateur signe et envoie le certificat d’attributs à l’utilisateur.

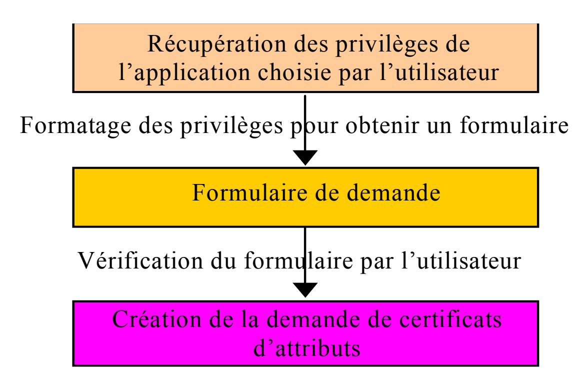


Figure 4-6 Scénario de création d’une demande de certificat d’attributs

La génération des certificats d’attributs en XML dans l’E-IGP se base essentiellement sur les *DTDs* (*Document Type Definition*) [W3CDTD]. La *DTD* est une grammaire permettant de vérifier la conformité du document *XML*. En effet, c'est grâce à celle-ci que l'administrateur personnalisera les certificats d'attributs que l'utilisateur pourra demander.

Les *DTDs* serviront donc de schémas pour la génération des certificats d'attributs en *XML*. Les données qu'elles contiennent sont extraites, de manière transparente pour l'utilisateur, par un *Servlet* [JAV] appelé par l'utilisateur (voir figure 4-7). Le formulaire de demande de certificat d'attributs est, par la suite, créé dynamiquement à partir des données contenues dans la *DTD* choisie. Une option de cette personnalisation permettra de vérifier que les fichiers *XML* créés (correspondant chacun à une demande de certificat d'attributs) soient bien valides, c’est-à-dire, qu'ils soient bien conformes à la *DTD* choisie par l'utilisateur.

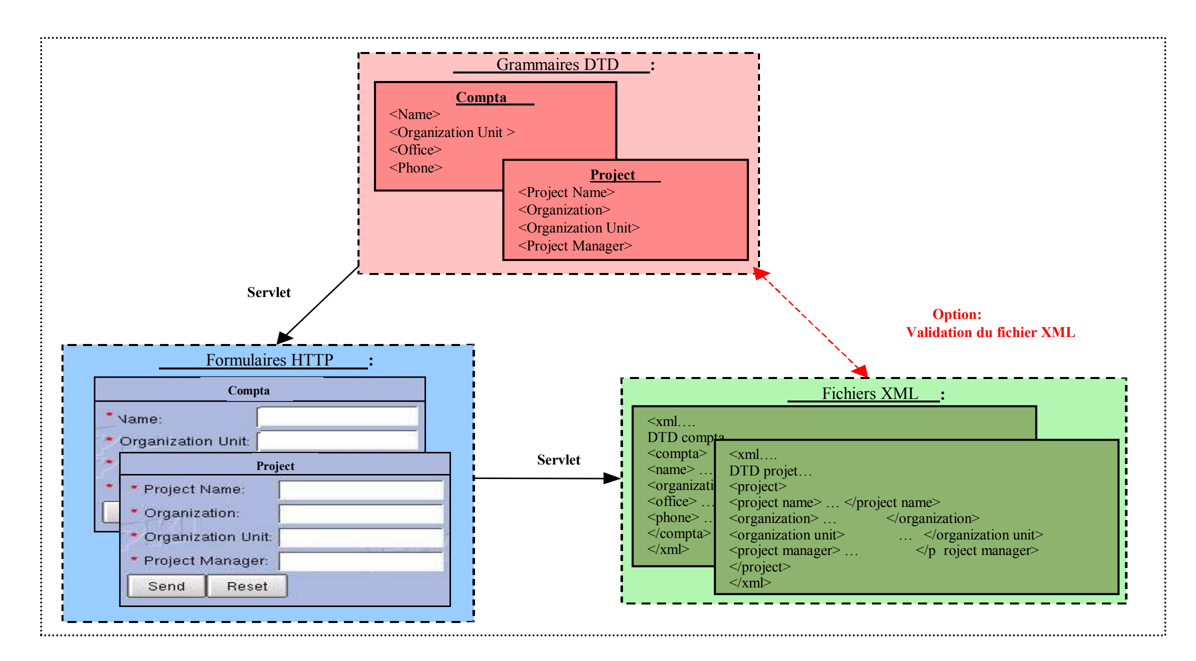


Figure 4-7 Rôle des *DTDs*

Un certificat d'attribut doit être couplé à un ou plusieurs certificats d'identité valides. Ce certificat d'attributs généré est signé par la clef privée de l'autorité *GCAX* (l'administrateur). Le format de signature utilisé est le format *XML-DSIG* (voir figure 4-11) [RFC2807], afin d'éviter la signature en format *PKCS#7* [RFC2315] qui est spécifiée en *ASN.1*.

XML-DSIG permet d'attacher une signature au certificat dans une étiquette, de rendre possible l'inclusion de la signature numérique dans le certificat lui-même, et même de pouvoir inclure le certificat dans l'étiquette de signature numérique.

Une fois ce certificat d'attributs créé, il est envoyé au client. Une trace de ce certificat d'attributs est sauvegardée dans la base de données commune entre les deux modules "*GCAX*" et "*V&CAA*".

4.3.1.1.1 Les privilèges variables des applications

La problématique étant de contrôler les droits des utilisateurs lors de leur accès aux applications réseau, nous avons choisi de définir des critères et des niveaux de privilèges que l'on peut donner aux utilisateurs pour chaque application.

Le niveau de privilèges le plus commun correspond à l'administration ou à la simple utilisation de l'application. Ensuite, les privilèges peuvent être très variés, par exemple, ils peuvent correspondre à l'aspect général de l'application, aux options de mise en page, etc.

Ces différents critères sont alors stockés dans une base de données sous forme de *DTD*. Le rôle d'une *DTD* est de définir la structure interne de certificat d'attributs. Ainsi, la *DTD* d'une application permet de structurer les droits qu'elle va accepter de la part des utilisateurs. Le document que ces derniers présenteront sera formaté pour les besoins de l'application.

Chaque application définit ses critères de privilèges variables dans un document qui est stocké dans une base de données de l'E-IGP.

La figure 4-8 représente un exemple de *DTD* pour une application de convertisseur de devises en ligne:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Convertisseur (Langue, Couleur, Droits) >
<!ATTLIST Convertisseur URL CDATA #FIXED
"http://localhost:8080/portail/index.jsp">

<!ELEMENT Langue (#PCDATA) >
<!ATTLIST Langue TypeData CDATA #FIXED "list">
<!ATTLIST Langue Values (Français | Anglais) "Français">
<!ATTLIST Langue Label CDATA #FIXED "Langue du convertisseur">

<!ELEMENT Couleur (#PCDATA) >
<!ATTLIST Couleur TypeData CDATA #FIXED "list">
<!ATTLIST Couleur Values (Bleu | Vert | Jaune) "Jaune">
<!ATTLIST Couleur Label CDATA #FIXED "Couleur du convertisseur">

<!ELEMENT Droits (#PCDATA) >
<!ATTLIST Droits TypeData CDATA #FIXED "list">
<!ATTLIST Droits Values (Utilisateur | Administrateur) "Utilisateur">
<!ATTLIST Droits Label CDATA #FIXED "Droits">
```

Figure 4-8 *DTD* d'un convertisseur de devices

Cet exemple, très simple, illustre la diversité des privilèges qui peuvent être accordés à un utilisateur.

4.3.1.1.2 L'encodage de la demande de droits en XML

La complexité du codage *ASN.1* des certificats d'attributs bien qu'adaptée à la gestion des droits, se révèle complexe à mettre en œuvre. Nous avons donc choisi un encodage en *XML* [SSGRR03].

Lorsque le formulaire est rempli par l'utilisateur, une demande de droits est créée à partir de ces informations. Cette demande est codée en *XML* et stockée dans la base de données des demandes en attente. L'intervention de l'administrateur de l'*IGP* est alors indispensable pour transformer cette demande en certificat d'attributs, qui sera ensuite utilisable auprès du portail d'accès aux applications (voir figure 4-9).

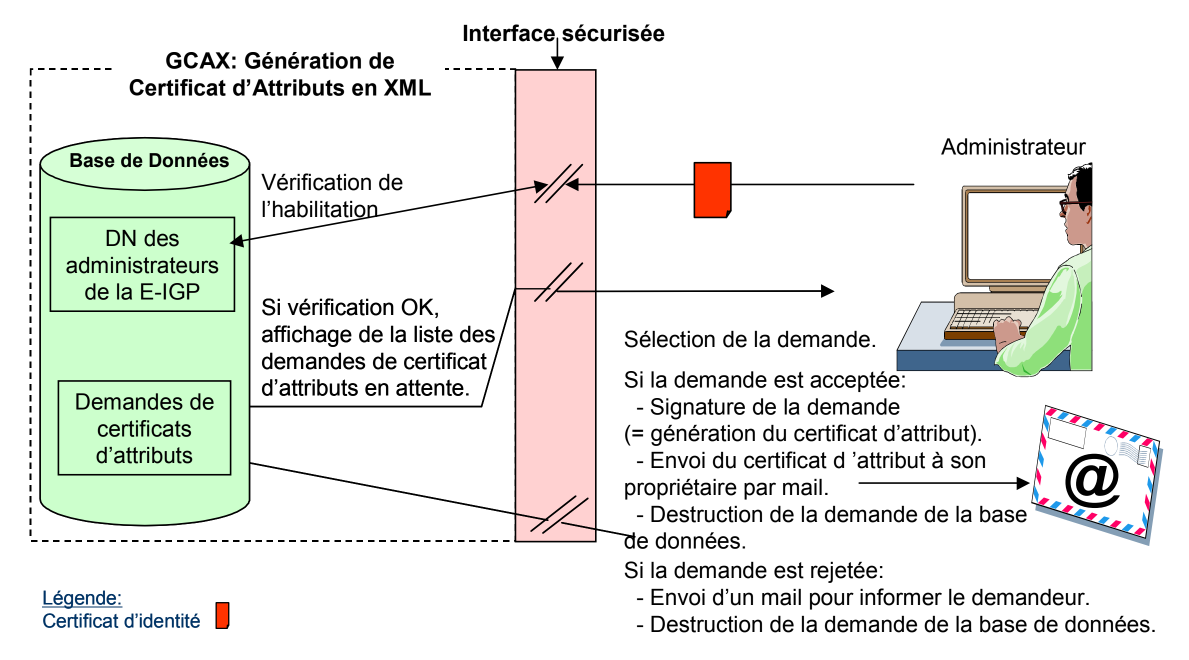


Figure 4-9 Validation des demandes de certificats d'attributs

4.3.1.1.3 Importance de XML

XML a pris une importance considérable dans le monde du commerce électronique puisqu'il autorise l'échange de données dont on définit la structure spécifique dans un langage standardisé.

Ainsi, l'approche de spécification des certificats d'attributs en *XML* bénéficie des avantages suivants relatifs au langage *XML* [SAR02]:

1. Ce standard promulgué par le *W3C* [W3C], définit un format universel de représentation des données. *XML* deviendra en outre le pivot de l'interopérabilité, en permettant l'échange de documents qui ne transitent pas forcément par Internet, tels que ceux générés par les différentes applications bureautiques.
2. A la manière d'*ASN.1*, *XML* est un langage extensible de description des données, structuré et indépendant des plates-formes. Un avantage par rapport à *ASN.1* est sa lisibilité (le fait qu'il soit en format de texte simple le rend lisible à la fois par les machines et par les utilisateurs. Ainsi, les certificats sont plus explicites pour les utilisateurs et les développeurs informatiques), mais il aurait été tout à fait possible de réaliser avec *ASN.1* ce qu'on fait avec *XML*. L'unicité des *OID* (*Object Identifier*) *ASN.1* est traitée en *XML* en faisant des références à des ressources uniques dans des domaines de nommage.
3. *XML* est un métalangage qui offre la possibilité de définir les balises dont on a besoin et de leur associer une interprétation. *XML* est flexible, ce qui limite les problèmes d'interopérabilité.
4. *XML* est un langage orienté objet qui définit sans ambiguïté aussi bien le contenu que la structure d'un document. Il fait la distinction entre la structure d'un document et sa présentation. Orienté vers le monde Internet et web, *XML* distingue la nature essentielle des données transportées (types, hiérarchie dans le document), de la manière dont l'environnement d'utilisation les traite et les affiche. Il est alors plus facile de créer des agents d'affichage qui sont adaptés aux terminaux et qui optimisent la visualisation des données. Les langages tels que *HTML* lient intimement les données et leurs fonctions d'affichage, ce qui rend difficile le portage de pages dans des environnements de visualisation contraints.
5. Il nécessite une puissance de calcul pour le traitement beaucoup plus faible que celle de l'*ASN.1* (car il est facile de parser un document *XML*, ce qui n'est pas le cas d'un document *ASN.1*) [OCTALIS].
6. *XML* peut être utilisé en parallèle avec les protocoles d'Internet, *DTD*s et d'autres utilitaires qui peuvent interpréter la sémantique de contenu d'un document *XML*. Les descriptions des structures de documents, les *DTD*, sont réutilisables et bénéficient, par principe, de certains concepts de la technologie objet. On peut réutiliser les structures des données et réduire ainsi les durées de développement des logiciels associés.
7. La possibilité de faire des références externes (localisation de documents sur le réseau) et croisées entre section de documents.

Les avantages potentiels de *XML* comme véhicule de communication sont significatifs. Ces propriétés font de *XML* un outil de choix pour, d'une part, encapsuler et véhiculer des données de transaction d'une manière souple et, d'autre part, les afficher sur les terminaux variés (ordinateurs, téléphones portables, assistants personnels, etc.). Mais véhiculer des données de transaction sur un réseau appelle automatiquement leur protection. La signature *XML* [RFC2807] [RFC3275] intervient à ce niveau.

4.3.1.1.4 La création du certificat d'attributs

L'administrateur de l'infrastructure de gestion des privilèges a le rôle d'autorité de confiance pour la signature des demandes de droits.

La consultation des demandes lui permet soit de les rejeter, et dans ce cas l'utilisateur est informé par mail, soit de les accepter.

La signature de la demande codée en *XML* va produire un certificat d'attributs (voir figure 4-10), lui aussi codé en *XML*.

La signature *XML* [RFC3275] [SAR02] de la demande a plusieurs avantages :

- Elle est basée sur *XML*, plus simple que celle basée sur le codage *ASN.1*.
- Elle permet de signer tout ou une partie du document ou même une partie extérieure référencée par une *URI* dans le document.
- Elle garantit l'intégrité du contenu du document et confirme l'identité de l'autorité signataire.
- Elle autorise également plusieurs entités à signer les différentes parties d'un même message sans pour autant invalider les autres [RFC2807].
- La signature est embarquée dans le document, «*envelopped signature*» [RFC3275].

Pour toutes ces raisons, nous avons choisi d'utiliser la signature *XML* des certificats d'attributs.

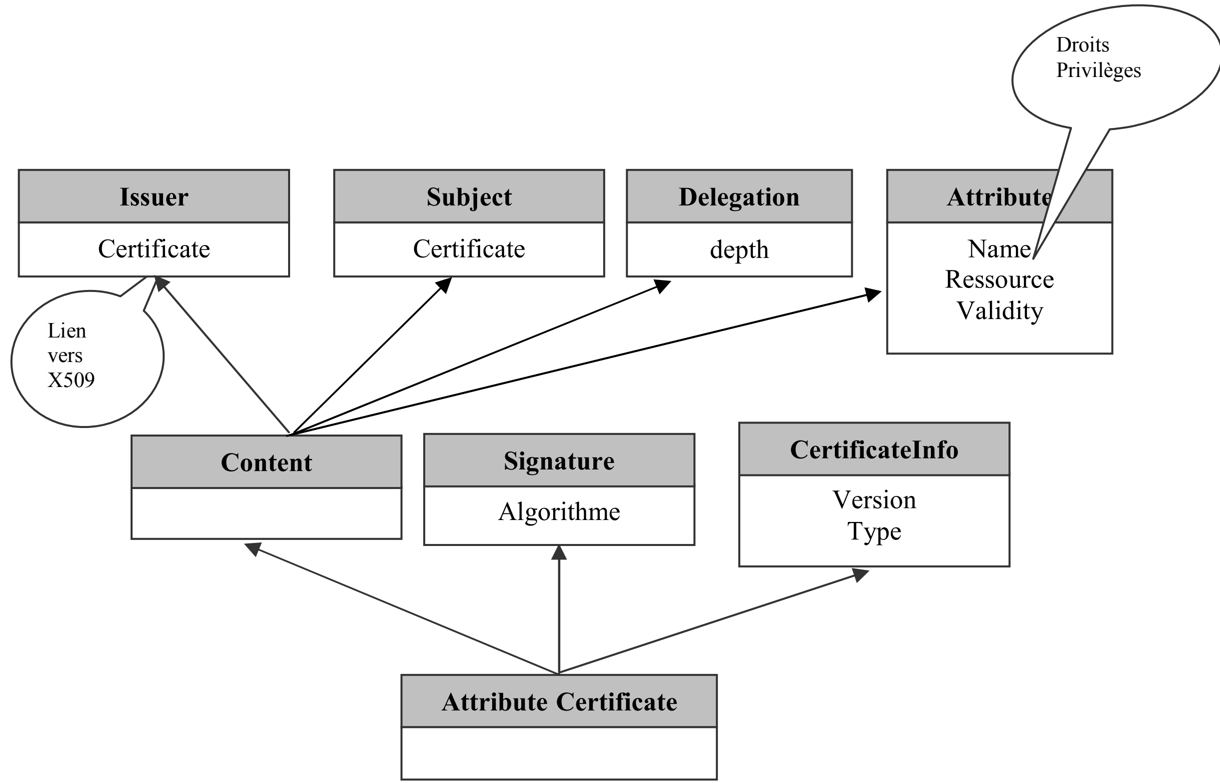


Figure 4-10 Format d'un certificat d'attributs en *XML*

Une fois le certificat d'attributs créé (voir figure 4-11), il est envoyé par mail à son propriétaire. L'utilisateur doit alors l'activer pour pouvoir en faire usage. Pour cela, il s'adresse au vérificateur de droits (*V&CAA* : Module de Vérification et de Contrôle d'Accès aux Applications).

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>[……]</ds:CanonicalizationMethod>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-
sha1"></ds:SignatureMethod>
     <ds:Reference URI="#IcarePaquet">
          <ds:Transforms>[……]</ds:Transforms>
          <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>PlUx3g+Sh92GDPCl+v+e760hyTs=</ds:DigestValue></ds:Reference></ds:SignedInfo>
<ds:SignatureValue>EhgqW6AxKZVJfu8HKm4HqsnO0MszMZP9UnRvyPGCwaepIdja/oroSSltfHOCJkar8JACEWMRdnd5W
0Nof81/h4t1uIhd/r7bVqgZYu27NLQ=</ds:SignatureValue>
<ds:KeyInfo>[……]</ds:KeyInfo>
<ds:Object Id="IcarePaquet">
<AttributeCertificate>
     <CertificateInfo><Version>V1.0</Version>
       <Type>AttributeCertificate</Type>    <Id>57</Id></CertificateInfo>
     <Content>
        <Issuer>
                <DN>EMAILADDRESS=demerjia@enst.fr, CN=JacquesDemerjian CA, O=ENST, L=PARIS</DN>
                <X509Data>MIICW[……]+EGHJjR</X509Data>
        </Issuer>
        <Holder>
           <Identity>
              <UserDN>EMAILADDRESS=thomas@icare1.tai, CN=SYLVAIN</UserDN>
              <Serial>1</Serial>
              <PublicKey>MIGfMA0[…….]AQAB</PublicKey>
           </Identity>
        </Holder>
        <Validity>
           <ValidityFrom>2003/06/25 10:18:14</ValidityFrom>
           <ValidityTo>2003/06/30 00:00:00</ValidityTo>
       </Validity>
        <Attribute>
           <Droits>Administrateur</Droits>
           <Couleur>Jaune</Couleur>
           <Langue>Français</Langue>
           <Application>Convertisseur Euros</Application>
           <Resource>http://localhost:8080/portail/index.jsp</Resource>
        </Attribute>
     </Content>
</AttributeCertificate>
</ds:Object>
</ds:Signature>
```

Figure 4-11 Exemple d'un certificat d'attributs en *XML signé* par *XMLDSIG*

4.3.1.2 V&CAA : Module de Vérification et de Contrôle d'Accès aux Applications

Le rôle *V&CAA* est d'activer le certificat d'attributs de l'utilisateur. Pour cela, l'utilisateur s'authentifie puis présente son certificat d'attributs.

Le *V&CAA* est une interface (voir figure 4-12) qui prend en charge la gestion de l'accès et l'utilisation sécurisée des applications (vérification de la signature, de la grammaire, permission et condition d'utilisation….). Cette interface utilise une *API* qui contient toutes les fonctions cryptographiques de base telles que le hachage, le chiffrement, la signature, etc.. L'avantage de cette interface réside dans sa portabilité et sa transparence pour tout type d'application.

Une fois le client récupère son certificat d'attributs, il peut s'en servir pour accéder et utiliser une application donnée.

Les étapes à suivre pour accéder à l'application sont alors:

1. Le client utilise le protocole *SSL* pour ouvrir une connexion sécurisée avec le serveur. A travers SSL, il s'authentifie en présentant son certificat d'identité X509.
2. Le client présente son certificat d'attributs pour accéder à l'application souhaitée.
3. *V&CAA* télécharge le certificat d'attributs sur son serveur, puis effectue une série de tests :

a) Vérification de la signature de l'autorité d'attributs du certificat d'attributs en *XMLDSIG* (voir paragraphe 4.3.1.1.4).

b) La conformité du certificat d'attributs à la *DTD* de l'application choisie par le client.

c) La vérification de la relation entre le certificat d'identité X.509 et le certificat d'attributs du client.

d) La vérification de la signataire du certificat d'attributs, reconnue comme autorité de confiance.

e) La vérification de la date de validité du certificat d'attributs.

f) L'extraction du certificat d'attributs des champs suivants (le rôle du client (administrateur,..); l'*URL* de l'application,..).

Les utilisateurs ont accès à l'application une fois l'authentification et l'autorisation sont accordées selon les privilèges. Les ressources sont présentées aux utilisateurs auxquelles ils peuvent avoir accès et utiliser, tandis que les fonctions d'application et ressources restreintes demeurent protégées. Dans le cas où une anomalie serait observée, l'utilisateur en est informé et l'application n'est pas lancée.

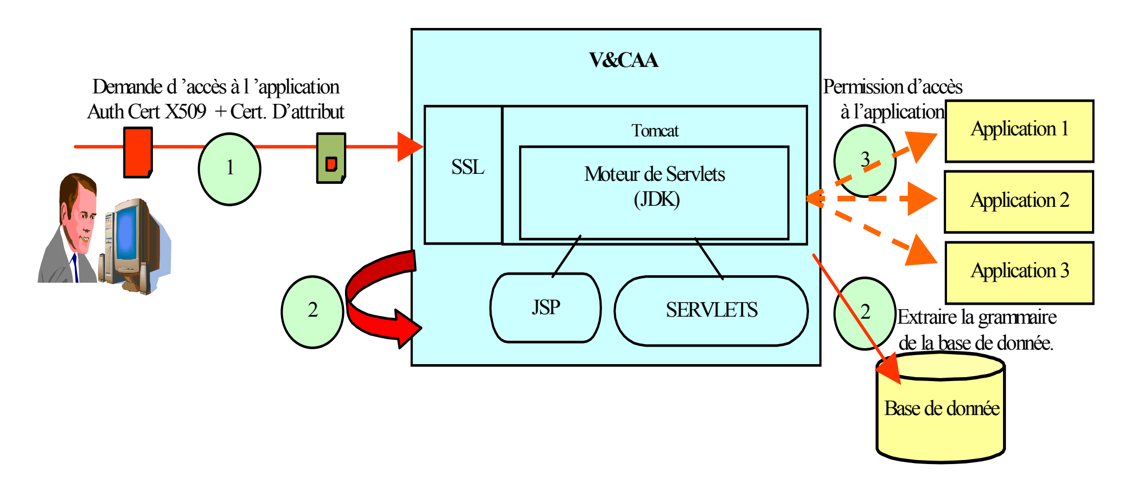


Figure 4-12 *V&CAA* : Vérification et Contrôle d'Accès aux Applications

4.3.2 Choix technologiques

Les choix technologiques que nous avons pris pour implémenter le *E-IGP* sont les suivants :

1. *Java*, *JSP et Servlet : Ces* techniques sont utilisées dans le développement des deux modules *GCAX* (Génération de Certificat d'Attributs en *XML*) et *V&CAA* (Vérification et de Contrôle d'Accès aux Applications). L'*API* cryptographiques de base de l'*E-IGP* est écrit en *Java* qui possède quelques avantages par rapport à d'autres langages comme *C* [RIT].

   Java est un langage objet permettant le développement d'applications complètes s'appuyant sur les structures de données classiques (tableaux, fichiers) et utilisant abondamment l'allocation dynamique de mémoire pour créer des objets en mémoire. La notion de structure est remplacée par la notion de classe au sens de la programmation objet. Le langage *Java* définit également une interface graphique (*GUI : Graphical User Interface*) facilitant le développement d'applications interactives et permettant à l'utilisateur de "piloter" son programme dans un ordre non imposé par le logiciel.

   Un programme *Java* est portable au sens où il peut s'exécuter sur des ordinateurs fonctionnant sous différents systèmes d'exploitation. Les programmes écrits en *Pascal* [PAS] ou en langage *C* sont aussi portables, à condition de compiler le code source sur le même type de machine que celle où le programme doit être installé et exécuté. *Java* est portable d'une plate-forme (matériel et

système d'exploitation) à une autre sans recompilation. Le compilateur produit un langage intermédiaire appelé "*bytecode*" qui est interprété sur les différentes machines. Il suffit alors de communiquer ce "*bytecode*" et de disposer d'un interpréteur de "*bytecode*" pour lancer l'exécution d'un programme *Java*. Les navigateurs du *Web* ont intégré un interpréteur de "*bytecode*" qui leur permet d'exécuter des programmes (*applets*) *Java*.

Cette portabilité est possible grâce à une normalisation indépendante de la plateforme. Cette portabilité est souvent résumée de manière un peu commerciale par « *write once, run anywhere* » (écrivez une fois, exécutez partout).

La richesse du langage *Java* tient aussi à sa bibliothèque de classes (*API : Application Programming Inteface*) qui permet de traiter des applications très diverses (interfaces graphiques, mathématiques, etc.). Là également, il suffit de disposer du "*bytecode*" des bibliothèques pour pouvoir les utiliser sur n'importe quel ordinateur.

L'architecture de la plate-forme *J2EE* de *Sun* spécifie la partie serveur de présentation. Cette gestion de la présentation de page *HTML* dynamique repose sur les *servlets* et les *JSP* (*JavaServerPages*). L'utilisation du langage *Java* rend cette solution indépendante du système d'exploitation, et il est ainsi possible de choisir l'environnement de développement le mieux adapté (*OS*, plate-forme).

Bien que principalement utilisées pour la couche de présentation, les *servlets* et les *JSP* peuvent servir à d'autres tâches. Dans un contexte *B-to-B*, elles peuvent être activées pour envoyer et recevoir des messages *XML*. Les *servlets* fournissent une infrastrucrure technique de base pour construire des applications *Java* dans un contexte *Web*, généralement associé au protocole *HTTP*. Si les *JSP* se révèlent être d'excellents outils pour construire la couche graphique des applications, en laissant toute liberté pour manipuler du code *Java*, il convient, pour satisfaire les règles de l'art et garantir scalabilité et maintenabilité, de dissocier dans l'architecture de l'application la simple mise en forme de l'information de la logique applicative.

2. *XML* et *XML* S*ignature :* Le module *GCAX* (Génération de Certificat d'Attributs en *XML)* crée des certificats d'attributs spécifiés en *XML,* puis les signes en utilisant *XML Signature* [RFC3275]. Le module *V&CAA* (Vérification et de Contrôle d'Accès aux Applications) vérifie la signature de l'autorité d'attributs du certificat d'attributs en utilisant *XML Signature.*

   Si le *HTML* et *HTTP* ont constitué ensemble le premier accélérateur du développement d'Internet, *XML* est le second. *XML* est une clef pour l'intégration tant à l'intérieur des entreprises qu'entre les entreprises. En tant que format universel d'échange, il a donné naissance à de nombreux protocoles (*XML Encryption* [*XENC*], *XML Signature* [XSIG] [RFC2807], *XKMS* [XKMS], *SOAP* [SOAP], etc.). Les techniques de transformation associées à *XML* sont utiles pour répondre à la diversification actuelle du poste client et à sa personnalisation (pour plus de détailles concernant nos choix techniques voir Annexe E).

4.4 Conclusion

Les différents types de certificats existants ne correspondent pas toujours aux besoins de l'utilisateur et/ou de l'application. A ce jour, les champs du certificat d'attributs spécifiés dans le standard existant sont fixés et ne correspondent pas toujours aux besoins de l'utilisateur et/ou de l'application. D'où la nécessité de trouver une méthode pour remplir les vrais besoins de l'application (en précisant les paramètres dont l'application a besoin) et de l'utilisateur (en précisant la valeur des paramètres de l'application, les délégations, les rôles, la validité du certificat, etc) en même temps.

Dans ce chapitre nous avons proposé et implémenté une Extension de l'Infrastructure de Gestion de Privilège (*E-IGP*). L'*E-IGP* propose une nouvelle approche de certification qui satisfait exactement les besoins de l'application et de l'utilisateur. L'extension que nous avons proposée permet à

l'utilisateur de personnaliser sa demande de certificat d'attributs. Cette approche consiste à définir et associer une grammaire (*DTD*) spécifique pour chaque application. Le rôle de cette grammaire est de préciser tous les paramètres dont l'application a besoin. Les *DTD* serviront de schémas pour la génération des certificats d'attributs.

L'utilisateur choisit alors le type de certificat d'attributs qui correspond le plus à ses besoins et sélectionne la *DTD* correspondante. En effet, c'est grâce à cette *DTD* que l'administrateur personnalisera les certificats d'attributs que l'utilisateur pourra demander.

La proposition *E-IGP* servira pour le renforcement de la sécurité de protocole d'attribution dynamique des adresses *DHCP* (*Dynamic Host Configuration Protocol*) [RFC2131] qui sera détaillé dans les chapitres suivants.

PARTIE II

Services d'autorisation et

Intégration au protocole d'attribution dynamique des adresses

Chapitre 5

5 Etude et analyse de protocole DHCPv4

Résumé

Le protocole DHCP (Dynamic Host Configuration Protocol), défini par le RFC 2131[RFC2131] n'est pas très ancien dans l'échelle de l'histoire des réseaux (octobre 1993). Il est basé sur le protocole BOOTP (Bootstrap Protocol) [RFC951], lequel était bien plus ancien. Le but de DHCP est de permettre la livraison d'informations de configuration à des machines connectées à un réseau IP. Ces informations consistent essentiellement en des adresses réseau réutilisables, et en des paramètres de configuration additionnels et des options.

Ce protocole est aujourd'hui très répandu, car il s'avère très pratique pour gérer de larges parcs de machines. Son usage est tout à fait indiqué dans les entreprises et les universités, tant pour affecter aux machines des adresses IP statiques que des adresses IP dynamiques. Il permet de centraliser complètement la gestion des ressources en adressage. Il est également utilisé, aujourd'hui, dans certaines des solutions ADSL du marché.

Si le succès du protocole DHCP est indiscutable, il faut avouer que la sécurité n'était pas une préoccupation majeure à l'époque où il a été mis au point, en 1993, puis amélioré, en 1997. De réels efforts de sécurisation des protocoles n'ont été initiés qu'à partir de 1999, le projet le plus connu étant le projet SSL (Secure Sockets Layer) [RFC2246] de Netscape, qui a donné naissance au célèbre protocole HTTPS (HTTP over SSL) [RFC2616].

Depuis 1997, plusieurs extensions visant à parer aux principales failles de sécurité de DHCP intrinsèques ont été proposées. Pour autant, aucune d'entre elles ne corrige tous les problèmes! Cette deuxième partie de notre thèse consiste à exposer notre nouvelle proposition d'extension: Extended DHCP [DEMERDraft].
Les résultats de cette deuxième partie de notre thèse ont été publiés dans un draft au sein de l'IETF [DEMERDraft] et dans plusieurs conférences [ICETE], [SEC], [ICTTA], [SAR04].

5.1 Introduction

Le protocole d'attribution dynamique des adresses Internet est nécessaire pour le fonctionnement d'un grand nombre de réseaux. Cela est dû notamment au manque d'adresses Internet qui ne permet pas leur attribution statique et au fait que la mobilité des équipements est plus adaptée à l'adressage dynamique.

Le protocole *DHCP* (*Dynamic Host Configuration Protocol*) [RFC2131] est ainsi au centre des architectures réseaux car il attribue les adresses IP pour permettre aux équipements de s'insérer dans le réseau. Les motivations qui ont conduit à la conception du protocole *DHCP* n'avaient pas pour objectif sa sécurisation. On observe dès lors, une série d'attaques potentielles sur ce protocole. Le protocole n'a pas prévu une authentification du client afin de lui attribuer une adresse IP. Un intrus peut ainsi facilement s'introduire dans le réseau en s'appropriant soit de l'identité, soit de l'adresse physique d'un client authentique. Ceci a conduit à la conception de nouveaux mécanismes pour renforcer la sécurité de ce protocole [RFC3118] [CBDA] [HOMDraft]. Mais ces contributions possèdent des limites et sont, de plus, vulnérables à certains types d'attaques.

Nous proposons dans cette seconde partie de notre rapport une extension de ce protocole afin de permettre un contrôle strict sur les équipements par une procédure d'authentification forte. La solution

proposée est basée sur la notion de certificat d'identité et d'attributs (présentés dans les chapitres 1 et 2). L'objectif de cette solution est d'attribuer à l'équipement un certificat d'attributs contenant l'adresse IP allouée dynamiquement. Cette extension de *DHCP* est possible car le champ d'options d'une trame *DHCP* prévoit l'implémentation de nouvelles options. Pour leur authentification au sein des architectures réseaux, les équipements peuvent faire preuve de l'authenticité de leur adresse IP en présentant leur certificat d'identification X.509 et le certificat d'attributs. Ainsi, le serveur *DHCP* est adossé à un serveur d'attribution de privilèges.

La partie concernant notre contribution sur *DHCP* est organisée comme suit. Dans ce chapitre, nous décrivons le protocole *DHCP,* son fonctionnement, les failles et problèmes de sécurité de *DHCP*. Ensuite, nous illustrons les solutions existantes pour renforcer la sécurité de *DHCP*, ainsi que leurs limites. Le chapitre suivant expose notre proposition appelée *E-DHCP*, son principe, ses conditions, ses scénarios et ses avantages. Dans le septième chapitre, nous modélisons *E-DHCP* en *UML* et nous présentons l'implémentation et les tests du protocole *E-DHCP*.

5.2 Présentation du protocole *DHCP*

DHCP signifie *Dynamic Host Configuration Protocol*. Il s'agit d'un protocole de configuration dynamique d'hôte qui permet d'allouer à la demande des adresses IP aux équipements se connectant au réseau. *DHCP* est un protocole de la couche application utilisant IP/UDP [RFC768]. C'est est une extension du protocole *BOOTP* (*Bootstrap Protocol*) [RFC951], l'enrichissant de nouvelles fonctionnalités, comme l'allocation dynamique d'adresses IP.

5.2.1 L'évolution vers *DHCP*

Le protocole *BOOTP* a été initialement conçu pour activer la configuration d'amorçage des stations de travail sans disque sur des systèmes plus anciens. Ce protocole était nécessaire car les clients de ce type possédaient des fonctions limitées de stockage des informations de configuration requises au cours de leurs processus d'amorçage respectifs utilisés pour démarrer un réseau et s'y connecter. Avec *BOOTP*, l'administrateur du réseau est obligé d'attribuer une adresse IP à chaque machine qui peut se connecter à un sous-réseau. Il en résulte un gaspillage d'adresses IP car, même une station qui se connecte rarement se voit attribuer une adresse IP. Pour pallier les différents inconvénients du protocole *BOOTP*, *DHCP* a été mise en place. Avec *DHCP*, l'adresse IP est attribuée dynamiquement, à partir d'une plage d'adresses (*pool d'address*) IP disponibles, par un serveur *DHCP* au démarrage. En effet, le protocole *DHCP* permet à l'administrateur de réseau de créer un ensemble d'adresses pouvant être attribuées aux clients. De cette façon, il peut déterminer des intervalles d'adresses IP. Il peut donc décider de donner un intervalle pour un segment et un autre intervalle pour un deuxième segment. Ainsi, la gestion du réseau en est simplifiée. Le protocole *BOOTP* ne permet pas cette possibilité.

Pour des raisons d'optimisation des ressources réseau, *DHCP* permet l'allocation d'une adresse IP pour une durée déterminée appelée bail ou *lease* en anglais. Les adresses IP sont délivrées avec une date de début et une date de fin de validité. Par conséquent, dans le temps, une adresse IP peut servir à désigner différents équipements. Si un équipement doit garder plus longtemps que le bail une adresse, il doit le renouveler périodiquement. Cette caractéristique du *DHCP* lui permet de garder en mémoire, pour une certaine période de temps, les adresses IP qu'il assigne aux clients, ainsi que les informations de l'équipement en question, cela entraîne donc moins de transferts lors d'une connexion. La période de temps est définit par l'administrateur du réseau. Le protocole *BOOTP*, quant à lui, n'a pas cette fonction. Les postes clients utiliseront donc leur adresse IP et les informations reçues sur la configuration jusqu'à ce qu'il soit redémarré leur postes.

5.2.2 Format des trames *DHCP*

La structure des trames *DHCP* est basée sur celle des trames du *BOOTP*. Comme la structure de trame est la même, un serveur *DHCP* peut être capable de répondre à une requête *BOOTP*. De plus, un client *DHCP* peut interagir avec un serveur *BOOTP*, dans la mesure où la configuration proposée par le serveur convient au client. Le format de la trame *DHCP* est le suivant :

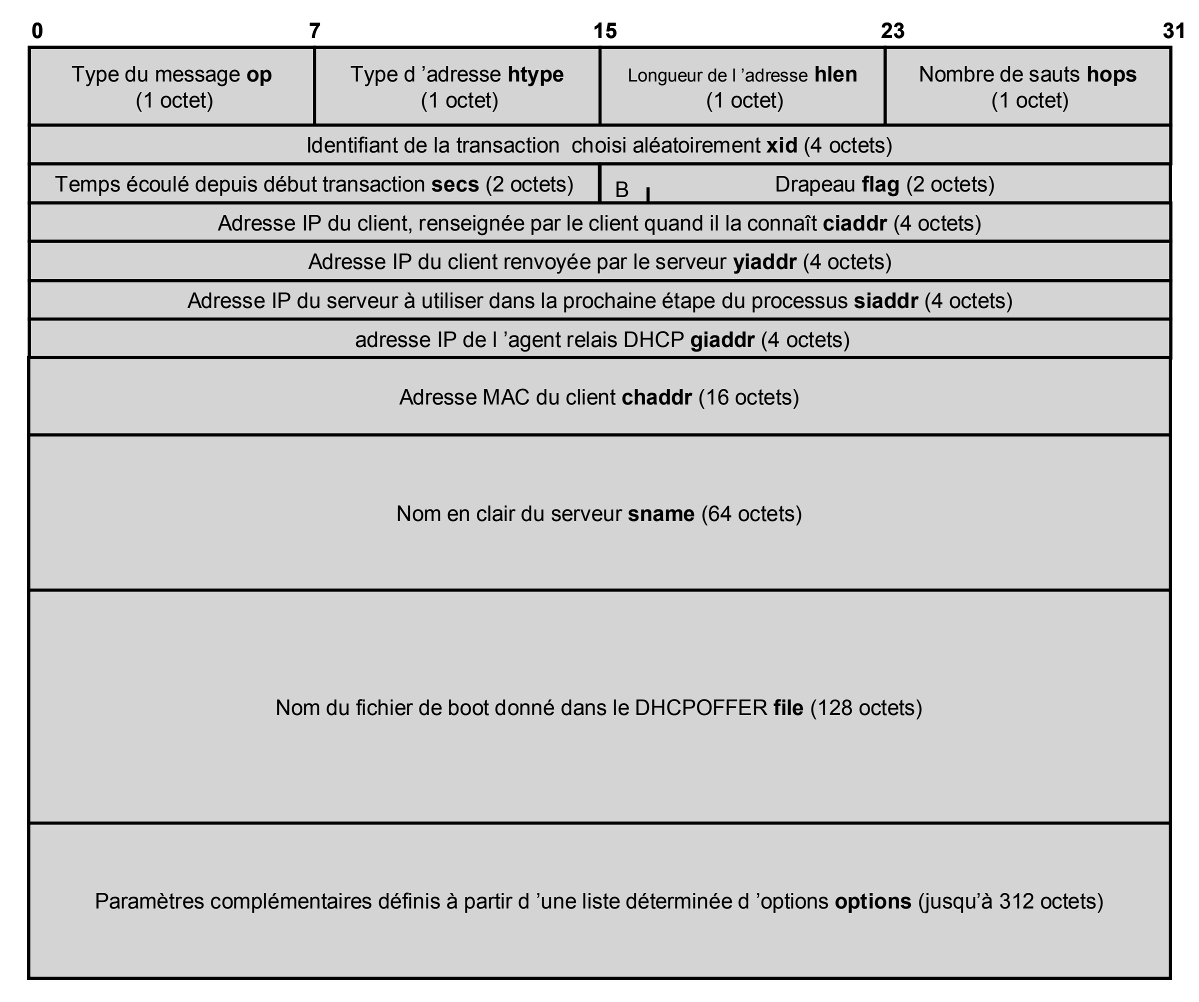


Figure 5-1 Format d'un paquet *DHCP*

Le trame *DHCP* est composé des champs suivants :

1. *"op"* est le type de message. Si op=1, le message est un *BootRequest* (trame *DHCP* émise par le client à destination du serveur). Si op=2, c'est un *BootReply* (trame *DHCP* émise par le serveur à destination du client).
2. *"htype"* définie le type de l'adresse MAC.
3. *"hlen"* est la longueur de l'adresse de l'interface matérielle.
4. *"hops"* est le nombre d'agents relais visités pour traiter le message. Le client *DHCP* assigne la valeur 0 à ce champ.
5. *"xid"* est l'identifiant de la transaction. Utilisé pour associer les requêtes d'un client et les réponses d'un serveur à une même transaction *DHCP*, il est choisi par le client *DHCP* et doit être unique sur le réseau local.

6. *"secs"* est le nombre de secondes depuis le début du processus de configuration. Le client augmente cette valeur lors de retransmission pour donner une indication de la durée du processus. Lorsque ce temps est grand, certains serveurs qui auraient ignoré la requête initialement, peuvent décider de fournir une réponse.
7. *"flags"* est le bit positionné le plus à gauche de ce champ et appelé le "*Flag de Broadcast*". Ce champ n'est pas utilisé dans *BOOTP*. Le bit le plus significatif est mis à 1 lorsque le client ne peut pas accepter un "*unicast*" avant la configuration complète de TCP/IP. Ainsi, le serveur *DHCP* sait qu'il doit faire un "*broadcast*" pour transmettre les réponses.
8. *"ciaddr"* est l'adresse IP du client. Ce champ est initialisé par le client seulement :
   - après que l'adresse ait été assignée avec succès,
   - lorsque le client renouvelle le délai d'expiration, ou
   - lorsque le client libère l'adresse obtenue précédemment.
9. *"yiaddr"* est l'adresse IP du client telle que fournie par le serveur. Ce champ est initialisé par le serveur lorsque celui-ci offre une adresse à un client ou confirme une adresse à un client.
10. *"siaddr"* est l'adresse IP du serveur à utiliser dans la prochaine étape du processus *BOOTP*. Ce champ est renseigné dans les messages *DHCPOFFER* et *DHCPACK* (voir figure 5-2).
11. *"giaddr"* est l'adresse IP de l'agent relais à utiliser.
12. *"chaddr"* est l'adresse de l'interface matérielle du client.
13. *"sname"* est le nom de domaine du serveur (optionnel), terminé par un nul (0x00).
14. *"file"* est le chemin d'accès complet du fichier de chargement de l'image système (optionnel), terminé par un nul (0x00).
15. *"options"* sont des paramètres optionnels utilisés pour fournir des informations de négociation entre le client et le serveur *DHCP*. Définies dans [RFC2132], ces options permettent de distribuer aux clients différentes informations en plus de leur adresse IP et leur masque de sous-réseau, telles que les adresses des serveurs *DNS*, *NTP* (*Network Time Protocol*) [RFC1305], *SMTP* (*Simple Mail Transfert Protocol*) [RFC2821], *IRC* (*Internet Relay Chat*) [RFC1459], etc. La longueur de ce champ dépend de la longueur maximale d'un message *DHCP*. Cette valeur peut être augmentée avec l'option "*Maximum DHCP Message Size*" (par défaut, 576 octets est la longueur minimale d'un datagramme IP).

La signification et le format des champs sont inchangés par rapport à *BOOTP*, à l'exception des champs "*flag*" et "*Options*" que l'on détaille ci-dessous :

Lorsque le bit *B est mis à 1*, du champ "*flag*" sur 2 octets, il indique que la réponse doit se faire en utilisant une adresse de diffusion, sinon, mis est à 0, en utilisant l'adresse MAC du client. Les autres bits ne sont pas employés et sont mis à 0. Certains clients *DHCP* ne sont pas capables de recevoir des datagrammes *unicast* avant que leur pile *TCP/IP* ne soit configurée. Dans ce cas, le client *DHCP* positionne à 1 le "*Flag de Broadcast*" (le bit le plus à gauche du champ "*flags*") sur toute trame *DHCP* qu'il émet. Un serveur *DHCP* recevant une telle trame répond alors systématiquement par *broadcast*. Si le "*Flag* de *Broadcast*" n'est pas positionné, les réponses des serveurs peuvent être émises sous forme *d'unicast*, l'adresse MAC destinatrice étant celle du champ "*chaddr*" existante dans la trame précédente du client et l'adresse IP destinatrice étant celle offerte par le serveur.

Le champ "*options*" correspond au champ "*vendor*" de *BOOTP*. Il a les mêmes fonctions, mais l'appellation est plus explicite. Voici quelques exemples d'options :

1."*DHCP MessageType*" : Type du message *DHCP*, champ impérativement renseigné dans toute trame *DHCP*. Par exemple, l'option type 1 correspond à un message *DHCPDISCOVER*.

2."*Server Identifier*" : Identifiant du serveur *DHCP*.

3."*Requested IP Address*" : Adresse IP souhaitée par le client *DHCP*.

4."*IP Address Lease Time*" : Durée de bail de l'adresse IP.

5."*Parameter Request List*" : Liste des paramètres IP attendus par le client.

6."*Option Overload*" : Indique que les champs "*sname*" et "*file*" vont être utilisés par les options *DHCP*.

7."*TFTP server name*" : Identifiant de serveur *TFTP* [RFC1350] quand le champ "*sname*" de l'entête *DHCP* est utilisé par les options *DHCP*.

8."*Bootfile name*" : Identifiant de fichier bootfile quand le champ "*file*" de l'entête *DHCP* est utilisé par les options *DHCP*.

9."*Message*" : Utilisé par le serveur *DHCP* pour fournir un message d'erreur au client dans le cas d'un échec.

10. "*Client-identifier*" : Utilisé par le client *DHCP* pour spécifier l'identificateur unique du client.

11. "*Vendor class identifier*" : Identification optionnelle du type de vendeur et de la configuration de client *DHCP*.

12. "*Maximum DHCP Message Size*" : Spécifie la longueur maximum acceptée d'un message *DHCP*.

13. "*Renewal (T1) Time Value*" : Spécifie l'intervalle de temps pour l'allocation de l'adresse jusqu'à la transition du client à l'état *RENEWING*.

14. "*Rebinding (T2) Time Value*" : Spécifie l'intervalle de temps pour l'allocation de l'adresse jusqu'à la transition du client à l'état *REBINDING*.

On notera que les options sont numérotées de 0 à 255. L'option "0" a une taille de 1 octet, et peut être utilisée pour signifier le début des options. L'option 255, elle aussi d'un octet signifie la fin des options. Les autres options peuvent avoir une longueur indéterminée. Elles commenceront toujours par un champ "*code*" suivi d'un champ "*longueur de l'option*". On notera que le calcul de la longueur de l'option n'inclut pas les champs "*code*" et "*longueur*". La procédure de définition et d'enregistrement d'une nouvelle option est détaillée dans les [RFC2132], [RFC2489] et [RFC2939].

5.2.3 Les messages *DHCP*

Les messages *DHCP* sont transmis via UDP. Bien que peu fiable, ce protocole suffit au transport des paquets simples sur réseau local. De facto, *DHCP* fonctionne aussi en mode non connecté. Le client *DHCP* n'utilise que le port 68 pour envoyer et recevoir ses messages de la même façon, le serveur *DHCP* envoie et reçoit ses messages sur un seul port, le port 67.

Plusieurs types de paquets *DHCP* ou messages sont susceptibles d'être émis soit par le client vers le ou les serveurs *DHCP*, soit par le serveur *DHCP* vers un client *DHCP*. Les messages échangés (voir figure 5-2) sont les suivants :

1.*DHCPDISCOVER* est utilisé par un client qui cherche sa configuration pour localiser le serveur.

2.*DHCPOFFER* permet à un serveur de proposer une configuration.

3.*DHCPREQUEST* permet à un client d'accepter l'offre de configuration d'un serveur et par conséquent, de refuser celle des autres serveurs.

4.*DHCPACK* permet à un serveur d'offrir une configuration.

5.*DHCPNACK* permet à un serveur de refuser d'offrir une configuration (par exemple, il ne possède plus d'adresses IP valables).

6.*DHCPDECLINE* permet à un client d'indiquer une offre de configuration invalide.

7.*DHCPRELEASE* permet à un client de libérer une offre de configuration précédemment acceptée.

Nom de message	Client -> Serveur	Serveur -> Client	Description d'usage
DISCOVER	x		Diffusé pour localiser les serveurs DHCP disponibles et pour demander une première configuration.
OFFER		x	Réponse à un message DISCOVER, qui contient l'offre des paramètres de configuration.
REQUEST	x		Message envoyé pour : (a) demander l'offre des paramètres proposés d'un serveur et refuser implicitement les offres venant des autres serveurs, (b) confirmer l'exactitude d'une adresse assignée précédemment (après, par exemple une réinitialisation de système), ou (c) prolonger le bail d'une adresse de réseau particulier.
ACK		x	Réponse à un message REQUEST, qui contient des paramètres et l'adresse IP du client.
NACK		x	Réponse pour signaler au client que son bail est expiré ou si le client annonce une configuration réseau incorrecte.
DECLINE	x		Annonce au serveur que l'adresse est déjà utilisée.
RELEASE	x		Le client libère son adresse IP et sa configuration courante.
INFORM	x		Le client demande seulement des paramètres locaux de configuration, il a déjà extérieurement configuré son adresse de réseau.

Figure 5-2 Les différents messages *DHCP*

5.3 Fonctionnement du protocole *DHCP*

DHCP fonctionne sur le modèle client-serveur : un serveur qui détient la politique d'attribution des configurations IP, envoie une configuration donnée pour une durée donnée à un client donné (typiquement, une machine qui vient de démarrer). Le serveur va servir de base pour toutes les requêtes *DHCP* (il les reçoit et y répond), aussi doit-il avoir une configuration IP fixe. Dans un réseau, on peut donc n'avoir qu'une seule machine avec adresse IP fixe : le serveur *DHCP*. Le protocole *DHCP* s'appuie entièrement sur *BOOTP* : il en reprend le mécanisme de base (ordre des requêtes, mais aussi le format des messages). *DHCP* est une extension de *BOOTP*.

Quand une machine vient de démarrer, elle n'a pas de configuration réseau (même pas de configuration par défaut), et pourtant, elle doit arriver à émettre un message sur le réseau pour qu'on lui donne une vraie configuration. La technique utilisée est le *broadcast* : pour trouver et dialoguer avec un serveur *DHCP*, la machine va simplement émettre un paquet spécial, dit de *broadcast*, sur l'adresse IP 255.255.255.255 et sur le réseau local. Ce paquet particulier va être reçu par toutes les machines connectées au réseau (particularité du *broadcast*). Lorsque le serveur *DHCP* reçoit ce paquet, il répond par un autre paquet de *broadcast* contenant toutes les informations requises pour la configuration. Si le client accepte la configuration, il renvoit un paquet pour informer le serveur qu'il garde les paramètres, sinon, il fait une nouvelle demande.

5.4 Les dialogues *DHCP* ou Scénario *DHCP*

Il existe deux cheminements pour la configuration d'un client par *DHCP*, selon que le client connaît ou pas son adresse IP. Ces deux cheminements sont illustrés par les deux figures 5-3 et 5-4.

5.4.1 1ère cas : Le client ne connaît pas son adresse IP

Dans le cas où le client *DHCP* connaît son adresse IP, l'allocation d'une adresse IP au client *DHCP* se déroule de la manière suivante (voir figure 5-3):

1. Le client *DHCP* diffuse, sur son réseau local physique, un message *DHCPDISCOVER* qui permet de découvrir les serveurs *DHCP* du réseau et de demander l'obtention d'une configuration *DHCP*. Le client peut éventuellement :

a) spécifier une liste de paramètres attendus pour sa configuration, en renseignant le champ option "*Parameter Request List*" dans sa trame de découverte,

b) suggérer des valeurs pour l'adresse réseau en renseignant le champ option "*Requested IP Address*" ou une durée de bail dans le champ option "*IP Address Lease Time*".

2. Chaque serveur, recevant le message *DHCPDISCOVER* et capable de satisfaire la demande du client *DHCP*, peut répondre avec un message *DHCPOFFER* qui inclut une adresse réseau IP valide dans le champ "*yiaddr*" (et d'autres paramètres de configuration des options *DHCP*). Le serveur n'a pas besoin de réserver l'adresse réseau offerte, bien que le protocole fonctionnera de manière optimale si le serveur évite d'allouer les adresses offertes à un autre client. Quand le serveur *DHCP* alloue une nouvelle adresse IP il doit vérifier que l'adresse réseau offerte n'est pas déjà utilisée, par exemple : le serveur devrait vérifier les adresses offertes par une requête d'écho ICMP. Un serveur *DHCP* devrait être implanté de manière à ce que les administrateurs réseaux puissent choisir de désactiver les vérifications des adresses nouvellement allouées. Le serveur *DHCP* transmet un message *DHCPOFFER* au client, en utilisant l'agent de relais *BOOTP* si nécessaire. Un serveur *DHCP* doit être en mesure d'attribuer une adresse IP différente de celle réclamée par le client. Il peut décider de ne pas attribuer d'adresse IP, même s'il reste des adresses libres, et ce pour des raisons administratives.

3. Le client peut recevoir 0 ou plusieurs messages *DHCPOFFER* d'un ou plusieurs serveurs *DHCP*. Il peut choisir d'attendre des offres multiples. Le client choisit une offre pour ses paramètres de configuration, basée sur la configuration présente dans le *DHCPOFFER*. Le client diffuse un message *DHCPREQUEST* qui doit inclure l'option "*Server Identifier*" indiquant quel serveur il a sélectionné et qui peut inclure d'autres options spécifiant les valeurs de configuration désirées. L'option "adresse IP demandée" doit être réglée sur la même valeur que "*yiaddr*" du message *DHCPOFFER* provenant du serveur *DHCP*. Ce *DHCPREQUEST* est diffusé et relayé au travers de l'agent de relais *DHCP*/*BOOTP*. Pour s'assurer que n'importe quel agent de relais fasse suivre le message *DHCPREQUEST* au même serveur *DHCP* qui à reçu le message *DHCPDISCOVER* original, le *DHCPREQUEST* doit utiliser les même valeurs dans l'en-tête du champ "*secs*" du message *DHCP* et être envoyé à la même adresse IP de diffusion que le message *DHCPDISCOVER* originel. Le client clôture à la fin d'un délai d'attente et retransmet le message *DHCPDISCOVER* si le client ne reçoit pas de messages *DHCPOFFER*.

4. Les serveurs *DHCP* reçoivent les diffusions *DHCPREQUEST* des clients. Les serveurs qui ne sont pas sélectionnés par le message *DHCPREQUEST* utilisent le message comme notification que le client décline leur offre. Le serveur *DHCP* sélectionné dans le message *DHCPREQUEST* engage une liaison pour le client dans sa mémoire permanente et répond avec un message *DHCPACK* qui contient la configuration pour le client demandeur. La combinaison entre "*Client-identifier*" ou "*chaddr*" et l'adresse réseau assignée constitue un identifiant unique pour le bail du client et sont utilisés à la fois par le client et le serveur *DHCP* pour identifier un bail auquel il sera fait référence dans tous les messages *DHCP*.

   N'importe quel paramètre de configuration dans le message *DHCPACK* ne devrait pas produire de conflit avec ceux du précédent message *DHCPOFFER* auquel le client répond. Le serveur ne devrait pas vérifier l'adresse réseau offerte a ce stade. Le "*yiaddr*" du *DHCPACK* est rempli avec l'adresse réseau sélectionnée. Si le serveur sélectionné est indisponible pour satisfaire au message *DHCPREQUEST* (par ex : l'adresse réseau demandée a été allouée), le serveur devrait répondre par un message *DHCPNAK*. Un serveur peut choisir de marquer comme indisponibles les adresses offertes aux clients dans un message *DHCPOFFER*. Le serveur devrait marquer comme disponible une adresse offerte à un client dans un message *DHCPOFFER* si le serveur ne reçoit pas de message *DHCPREQUEST* de ce client.

5. Le client reçoit un message *DHCPACK* avec les paramètres de configuration. Le client devrait faire une vérification finale sur les paramètres (par exemple, en utilisant le protocole ARP (*Address Resolution Protocol*) [RFC826] pour l'allocation de l'adresse réseau), et noter la durée du bail spécifié dans le *DHCPACK*. A ce moment, le client est configuré. Si le client détecte que l'adresse est déjà utilisée (par exemple, via l'utilisation de ARP) le client doit envoyer un *DHCPDECLINE* au serveur *DHCP* et relancer le processus de configuration. Le client devrait attendre un minimum de dix secondes avant de relancer la configuration pour éviter un trafic réseau excessif dans le cas d'un bouclage. Si le client reçoit un *DHCPNACK*, le client relance le processus de configuration. Le client clôture à la fin d'un délai d'attente et retransmet le message *DHCPREQUEST* si le client ne reçoit ni *DHCPACK* ni *DHCPNACK*. Le client retransmet le *DHCPREQUEST* conformément à l'algorithme de retransmission. Le client devrait choisir de retransmettre le *DHCPREQUEST* suffisamment de fois pour espérer contacter le serveur sans faire en sorte que le client n'attende trop longtemps avant d'abandonner ; par exemple : un client pourrait retransmettre le *DHCPREQUEST* 4 fois sur un total de 60 secondes, avant de relancer la procédure d'initialisation. Si le client ne reçoit ni *DHCPACK* ni *DHCPNAK* après l'utilisation de l'algorithme de retransmission, il retourne à l'état *INIT* et relance le processus d'initialisation. Le client devrait notifier à l'utilisateur que le processus d'initialisation n'a pas abouti et qu'il est relancé.

6. Le client peut choisir de renoncer au bail sur une adresse réseau en envoyant un *DHCPRELEASE* au serveur. Le client identifie le bail qu'il libère avec son "*Client-identifier*" ou "*chaddr*" et l'adresse réseau dans le message *DHCPRELEASE*. Si le client utilise un "*Client-identifier*" quand il obtient le bail il doit utiliser le même "*Client-identifier*" dans le message *DHCPRELEASE*.

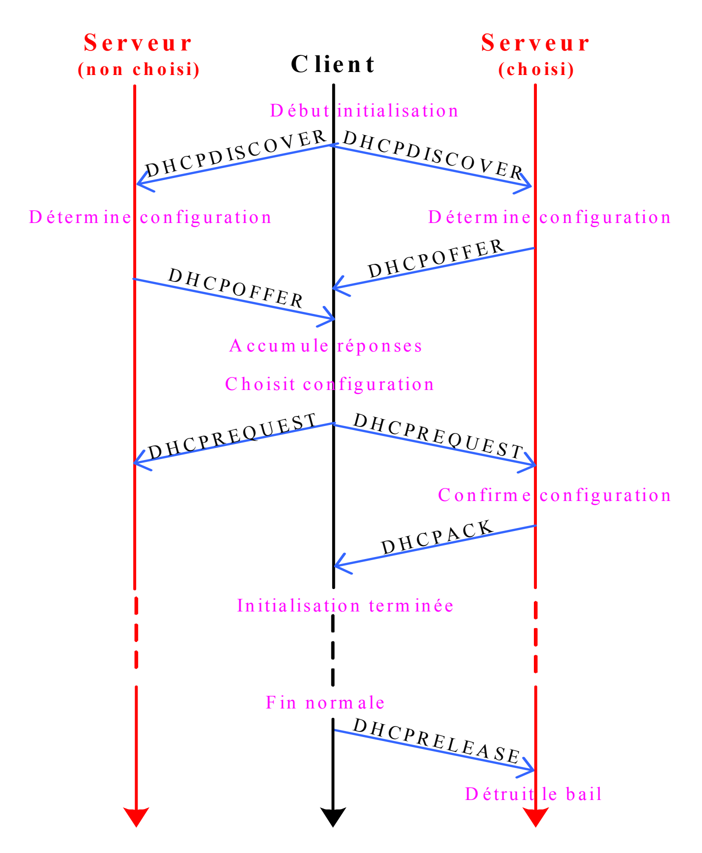


Figure 5-3 Scénario d'allocation d'une adresse IP au client *DHCP* qui connaît son adresse IP.

5.4.2 2ème cas : Le client connaît son adresse IP

Dans le cas où le client *DHCP* ne connaît pas son adresse IP, Le client peut également demander à ce que le serveur lui réattribue une adresse IP (voir figure 5-4). Dans ce cas la communication se fait de la façon suivante :

1. Le client *DHCP* diffuse un message *DHCPREQUEST* sur son sous réseau local. Le message inclut l'adresse réseau du client dans l'option "adresse IP demandée". Comme le client n'a pas encore reçu son adresse réseau, il ne doit pas remplir le champ "*ciaddr*". Les agents de relais *BOOTP* transmettent le message au serveur *DHCP* qui n'est pas sur le même sous réseau. Si le client utilise un "*Client-identifier*" pour obtenir son adresse, le client doit utiliser le même "*Client-identifier*" dans le message *DHCPREQUEST*.

2. Les serveurs *DHCP* qui connaissent les paramètres de configuration du client *DHCP* répondent avec un *DHCPACK* au client. Les serveurs ne devraient pas vérifier que l'adresse réseau du client est déjà utilisée ; le client peut répondre à une demande d'écho ICMP à cet instant. Si "*giaddr*" est 0x0 dans le message *DHCPREQUEST*, le client est sur le même sous réseau que le serveur. Le serveur doit diffuser le message *DHCPNAK* à l'adresse de diffusion 0xffffffff parce que le client peut ne pas avoir d'adresse réseau correcte ou le bon masque de sous réseau, et le client ne peut pas répondre à une requête ARP. De plus, le serveur doit envoyer le message *DHCPNAK* à l'adresse de l'agent de relais *BOOTP*, comme enregistré dans le "*giaddr*". L'agent de relais pourra, en retour, faire suivre le message directement, à l'adresse matérielle du client, de manière à ce que le *DHCPNAK* puisse être délivré même si le client s'est déplacé vers un nouveau réseau,

3. Le client reçoit le message *DHCPACK* avec les paramètres de configuration. Le client effectue une vérification finale sur les paramètres, et note la durée du bail spécifiée dans "*Client-identifier*" ou "*chaddr*" et l'adresse réseau. A ce stade, le client est configuré. Si le client détecte que l'adresse IP dans le message *DHCPACK* est déjà utilisée, le client doit envoyer un message *DHCPDECLINE* au serveur et relancer le processus de configuration en faisant la requête d'une nouvelle adresse réseau. Si le client reçoit un message *DHCPNAK*, il ne peut réutiliser l'adresse dont il se souvient. Il doit faire la requête d'une nouvelle adresse en relançant le processus de configuration, cette fois en utilisant la procédure (non abrégée) décrite dans la section 5.4.1. Cette action correspond également à un client qui se place dans l'état *INIT* du diagramme *DHCP*. Le client clôture à la fin d'un délai d'attente et retransmet le message *DHCPREQUEST* si le client ne reçoit ni un *DHCPACK* ni un *DHCPNAK*. Le client retransmet le *DHCPREQUEST* en accord avec l'algorithme de retransmission. Le client devrait choisir de retransmettre le *DHCPREQUEST* suffisamment de fois pour espérer contacter le serveur sans faire en sorte que le client (et l'utilisateur du client) n'attende trop longtemps avant d'abandonner ; par exemple : un client pourrait retransmettre le *DHCPREQUEST* 4 fois sur un total de 60 secondes, avant de relancer la procédure d'initialisation. Si le client ne reçoit ni *DHCPACK* ni *DHCPNAK* après avoir employé l'algorithme de retransmission, le client peut choisir d'utiliser une adresse réseau alloué précédemment et les paramètres de configurations pour les baux non expirés,

4. Le client peut choisir de renoncer à son bail sur une adresse réseau en envoyant un message *DHCPRELEASE* au serveur. Le client identifie le bail dont il veut se défaire avec "*Client-identifier*" ou "*chaddr*" et l'adresse réseau dans le message *DHCPREALEASE*. Notons que dans ce cas, le client retient son adresse réseau localement, normalement le client ne renonce pas à son bail lors d'un arrêt normal. Uniquement dans le cas ou le client doit explicitement renoncer à son bail, par exemple : le client va être déplacé sur un autre sous réseau, le client enverra un message *DHCPRELEASE*.

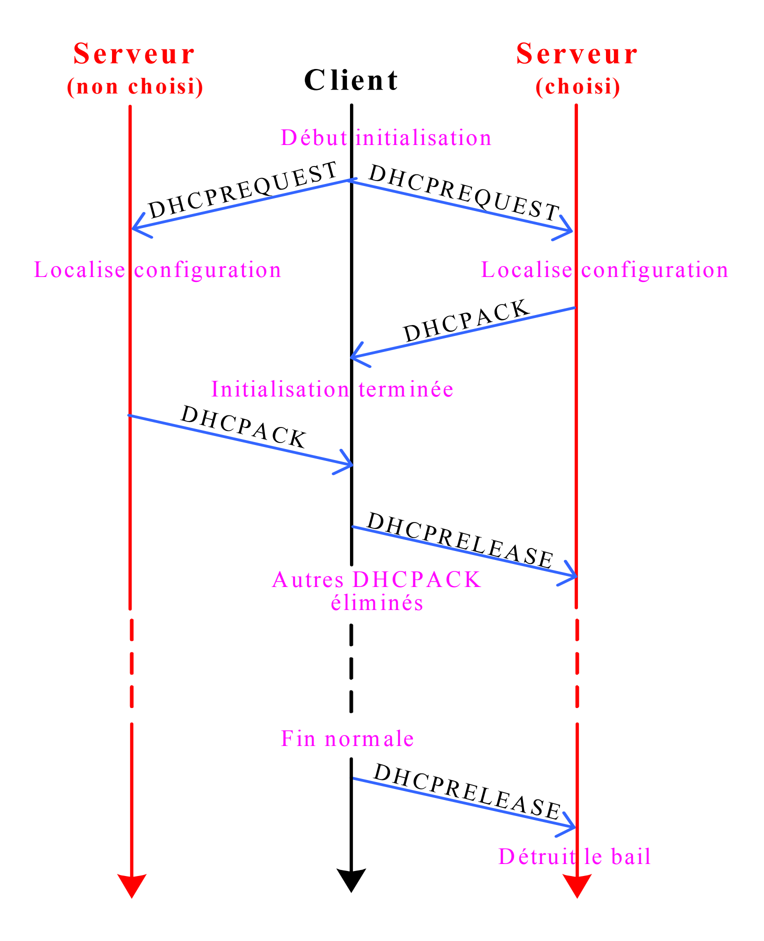


Figure 5-4 Scénario d'allocation d'une adresse IP au client *DHCP* qui ne connaît pas son adresse IP.

5.5 Problème de sécurité dans *DHCP*

Le protocole *DHCP* a été conçu dès le départ pour faciliter l'administration de réseaux, sans tenir compte des problèmes de sécurité que cela peut engendrer. L'extrême facilité d'utilisation qu'il procure explique son déploiement et son utilisation quasi systématique pour l'administration de parcs informatiques.

Les motivations qui ont conduit la conception du protocole *DHCP* n'avaient pas pour objectifs sa sécurisation. Il n'y avait aucune tentative dans la conception du protocole *DHCP* de mettre en place des systèmes de protection contre les hôtes d'Internet malveillants. On observe dès lors une série d'attaques potentielles sur ce protocole. Pour résumer les problèmes de *DHCP*, nous pouvons dire que les serveurs *DHCP* sont sujets aux attaques par déni de service, aux attaques de type "*man-in-the-middle*", de remplacement par de faux serveurs sur le réseau. Quant aux clients, s'ils sont également sujets à ces attaques, ils peuvent aussi être victimes de vol de service et d'usurpation d'identité. De plus, certaines attaques peuvent prendre plusieurs formes et avoir plusieurs cibles, comme les "*Denial of Service*" (épuisement d'adresse, saturation de la bande passante, déconnections des clients, etc.). Les nouvelles technologies de réseau sans fil facilitent encore le travail des pirates et externalisent le danger. Nous pouvons résumer les failles du protocole *DHCP* comme suit :

1. Déni de service :

   a) Refuser de configurer le client : Celui qui refuse de configurer le client peut être un serveur malicieux qui ignore les clients ou donne de mauvaises informations dans le but de ne pas répondre aux clients.

b) *Flood* : Un client *DHCP* peut inonder "*flooder*" le réseau en envoyant des requêtes continuellement pour saturer la bande passante,

c) Usurpation des clients: Ceci inclut les clients qui envoient des demandes erronées ou des clients malicieux qui se connectent pour épuiser les adresses, ou consommer des ressources du serveur *DHCP* ou du réseau,

2. Vol de service : Instanciation d'un client non autorisé afin d'accéder à des ressources ou des services du réseau. Le "vol de service" signifie la prise d'une adresse inutilisée pour l'accès au réseau ou l'usage d'une adresse assignée n'appartenant pas au client, contrairement à l'usurpation d'identité "*client masquerading*" d'un client *DHCP* qui se réfère spécifiquement à l'utilisation du champ *"chaddr"* d'un client légitime ou de l'identifiant du client.

3. Mauvaise configuration des clients : Elle résulte de l'action des serveurs malicieux qui donnent de mauvaises informations de configuration, ou encore les relais ou d'autres éléments qui modifient les paquets entre le client et le serveur. Le résultat est une mauvaise configuration (par exemple de faux serveurs *DNS* ou une fausse passerelle) des clients. Ceci peut entraîner des attaques potentiellement plus graves en redirigeant le trafic à travers un faux serveur, une interception ou une redirection de trafic.

4. Insertion, modification ou vol de paquets : Si un client, un serveur ou un relais est susceptible de planter ou de rebooter, au cas où des paquets d'un certain type sont envoyés, ceci pourrait simplement être un autre type de déni de service. De même, supprimer certains types de paquets pourrait conduire à l'expiration du bail d'un client, soit encore au déni du service. Un relais malicieux ou tout autre hôte peut également utiliser l'insertion et la suppression de paquets pour interrompre le service. La modification des paquets lors des échanges *DHCP* peut être employée pour mener différents types d'attaques sur le client ou le serveur,

Les problèmes de sécurité du protocole *DHCP* viennent du fait qu'il n'y a aucun mécanisme d'authentification des entités *DHCP* (client/serveur) et des messages *DHCP*. Le protocole *DHCP* ne supporte pas le mécanisme avec lequel les clients et les serveurs s'authentifient. De plus, le protocole *DHCP* n'assure pas l'intégrité des données échangées, ni leur confidentialité. Le serveur *DHCP* n'a aucun mécanisme de contrôle d'accès. Ainsi, n'importe quel client peut demander une adresse IP d'un serveur *DHCP* et n'importe quel intrus peut imiter un serveur *DHCP* pour envoyer des informations incorrectes à une demande de client *DHCP*.

5.6 Solutions existantes pour renforcer la sécurité de *DHCP*

Des solutions qui permettent de sécuriser le protocole *DHCP* existent, mais ne sont pas encore opérationnelles [RFC3118][HOMDraft][CBDA]. Dans cette partie, nous présentons les différentes solutions qui ont tenté de renforcer la sécurité de protocole *DHCP* afin d'assurer l'authentification d'entité *DCHP* (client et/ou serveur) et/ou des contenus des messages *DHCP*.

5.6.1 Authentification des clients *DHCP* par leurs adresses MAC

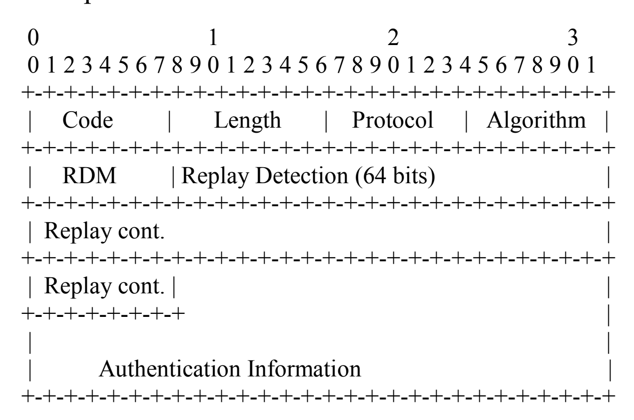
Pour renforcer la sécurité du protocole *DHCP*, un mécanisme d'authentification des clients *DHCP* par leurs adresses MAC a été proposé [WirNet]. Ce mécanisme peut être brièvement expliqué comme suit. À l'avance, les utilisateurs enregistrent l'adresse MAC de leurs terminaux sur le serveur *DHCP* avant d'accéder au réseau par le biais de l'administrateur réseau. Le serveur *DHCP* fournit par la suite une adresse IP uniquement à ces terminaux. Le serveur *DHCP* garde une comptabilité stricte des adresses MAC légales et fournit seulement le service à ces adresses [WirNet].

Ce mécanisme authentifie l'adresse MAC, donc la machine, mais pas le client. L'accès au réseau pour l'utilisateur est alors limité au terminal dont l'adresse MAC est enregistrée dans la base de données du serveur *DHCP*. D'autre part, l'intrus qui crée une adresse MAC légitime peut facilement tromper le serveur *DHCP* pour obtenir une adresse IP. De ce fait, le mécanisme d'authentification par adresses MAC ne résout pas les problèmes liés à la vulnérabilité de *DHCP*, d'où la nécessité de nouveaux mécanismes d'authentification plus solides.

5.6.2 Authentification des messages *DHCP*

[RFC3118] introduit une nouvelle option de *DHCP* (voir figure 5-5) dont le mécanisme peut fournir à la fois l'authentification de l'entité et l'authentification du message *DHCP*.

Les champs de la nouvelle option sont :



```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |    Length     |   Protocol    |   Algorithm   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     RDM       | Replay Detection (64 bits)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Replay cont.                                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Replay cont. |                                               |
+-+-+-+-+-+-+-+-+                                               |
|                                                               |
|           Authentication Information                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5-5 Option du RFC 3118

Avec respectivement :

- *Code* : Le code de l'option (0x5A) pour l'authentification.
- *Length* : La longueur de l'option.
- *Protocol* : Le code du protocole utilisé.
- *Algorithm* : L'algorithme utilisé dans le protocole précédemment cité.
- *RDM* : La méthode de détection de tentatives de rejeu *RDM* (*Replay Detection Method*).
- *Replay cont.* : L'information pour le champ précédent.
- *Authentication Information* : L'information utilisée pour l'authentification (taille variable).

Cette option peut utiliser deux techniques de protocole d'authentification à savoir, *"Configuration Token"* et *"Delayed Authentication"* [RFC3118].

5.6.2.1 Token Authentication

Cette méthode exige d'envoyer un mot de passe *"token"* du client *DHCP* au serveur *DHCP* et ceci, pour identifier le client *DHCP*. Le client *DHCP* insère le mot de passe "en clair" dans le message *DHCP* et le serveur *DHCP* fait la correspondance entre le mot de passe qu'il partage avec le client et celui contenu dans le message (reçu). Si le mot de passe reçu n'est pas le même que celui partagé entre le client et le serveur, le serveur rejette le message.

Cette méthode fournit seulement une authentification faible du client sans aucune authentification du serveur ni du message *DHCP*. Il est évident que ce mécanisme ne résout en rien les problèmes précédemment présentés, car ce type de protocole ne protège en rien la capture de ce secret (mot de passe), et ne protège pas contre l'intégrité des messages. Une fois une trame capturée, toutes les

attaques précédemment citées peuvent être utilisées encore. Le déni de service (*DoS*) est encore susceptible de fonctionner même sans le secret partagé divulgué. Ce protocole est vulnérable à l'interception des messages échangés entre clients et serveurs *DHCP*. La sécurité apportée par ce mécanisme est tellement basique qu'il est fortement recommandé de ne pas l'utiliser.

5.6.2.2 Delayed Authentication

C'est la méthode la plus sécurisée et la plus intéressante dans [RFC3118]. Elle fournit une authentification des entités (client et serveur) et des messages *DHCP* en même temps. La technique utilisée nécessite le partage d'une clef secrète différente entre chaque client et chaque serveur *DHCP* avec lequel le client souhaite utiliser l'option *"Delayed Authentication"*. Cette clef secrète est utilisée par l'émetteur (client ou serveur) pour calculer le *MAC (Message Authentication Code)* en utilisant l'algorithme *HMAC* [RFC2104] et la fonction de hachage *MD5* [RFC1321]. Tous les champs du message *DHCP*, y compris l'en-tête et les champs *options*, sont utilisés comme paramètres pour la fonction de calcul *HMAC-MD5* [RFC2202]. L'émetteur souhaitant utiliser cette option encapsule dans ces messages:

1. la valeur de *MAC* dans le champ *"Authentication Information",* et
2. l'identificateur de la clef secrète dans le champ *"secret ID"*.

Notons que chaque clef secrète possède un identificateur unique. Celui-ci sera récupéré du champ *"secret ID"* par le récepteur et utilisé pour extraire la clef secrète à partir d'une liste existante. Cette clef sera utilisée par le récepteur pour générer le *MAC* du message *DHCP* afin d'être comparé avec le contenu du champ *"Authentication Information"*. Si le *MAC* calculé par le récepteur n'est pas égal au *MAC* contenu dans l'option d'authentification, le récepteur doit rejeter le message *DHCP*. Si les deux *MACs* sont identiques, le message est authentifié et conservé.

5.6.2.2.1 Fonctionnement du mécanisme Delayed Authentication :

Voici les termes que nous emploierons dans cette section pour décrire le fonctionnement de ce mécanisme :

- *K* : la clef partagée connue uniquement du serveur et du client.
- *SID* : l'identifiant secret servant à identifier *K*.
- *KMAC* : haché signé avec la clef *K*.
- *Replay* : données anti-rejeu.

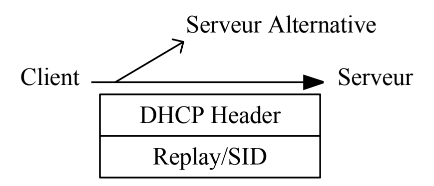


Figure 5-6 *DCHPDISCOVER*

Dans le datagramme *DHCPDISCOVER* (voir figure 5-6), le client *DHCP* fait une diffusion de sa demande en spécifiant qu'il souhaite s'authentifier avec le mécanisme *Delayed Authentication*. Le serveur *DHCP* sait quelle clef utiliser grâce au *SID* transmis.

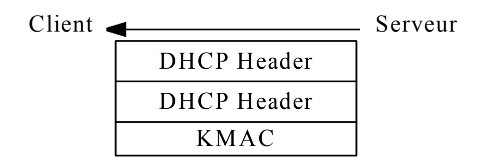


Figure 5-7 *DHCPOFFER*

Dans le datagramme *DHCPOFFER* (voir figure 5-7) le serveur *DHCP* répond avec une offre même si le client *DHCP* n'a pas été authentifié. Pour prouver son identité, le serveur *DHCP* calcule le *MAC* en utilisant l'algorithme de génération *HMAC* et la fonction de hachage *MD5*. Tous les champs du message *DHCP,* y compris l'en-tête et les champs options, sont utilisés comme paramètres pour la fonction de calcul *HMAC-MD5*. Le serveur *DHCP* inclut la valeur de *HMAC-MD5* (la valeur de *HMAC-MD5* appliquée sur le datagramme en utilsant la clef *K* référencée par *SID*) dans le champ *HMAC-MD5*.

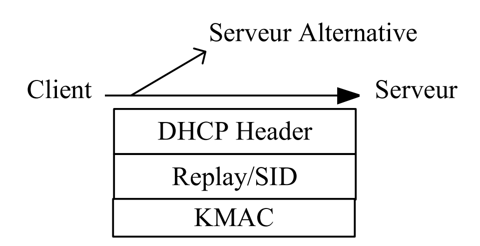


Figure 5-8 *DHCPREQUEST*

Dans le message *DHCPREQUEST* (voir figure 5-8), le client *DHCP* fait son choix d'offre et confirme son identité en appliquant la fonction *HMAC-MD5* (en utilsant la même clef secrète) sur le datagramme à envoyer, il prouve ainsi qu'il connaît *K*. Ce paquet est aussi envoyé aux autres serveurs *DHCP* desquels il avait reçu une offre.

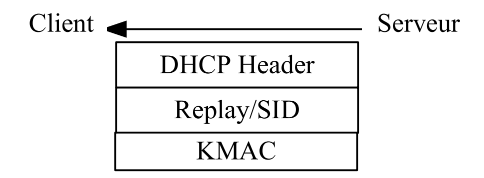


Figure 5-9 *DHCPACK*

Dans le message d'acquittement *DHCPACK* (voir figure 5-9), le serveur *DHCP* applique la fonction *HMAC-MD5* sur le message et valide son offre, le client *DHCP* peut alors utiliser sa nouvelle configuration.

Le format de l'option dans les messages *DHCPOFFER*, *DHCPREQUEST*, *DHCPACK* du protocole *Delayed authentication* est le suivant (voir figure 5-10):

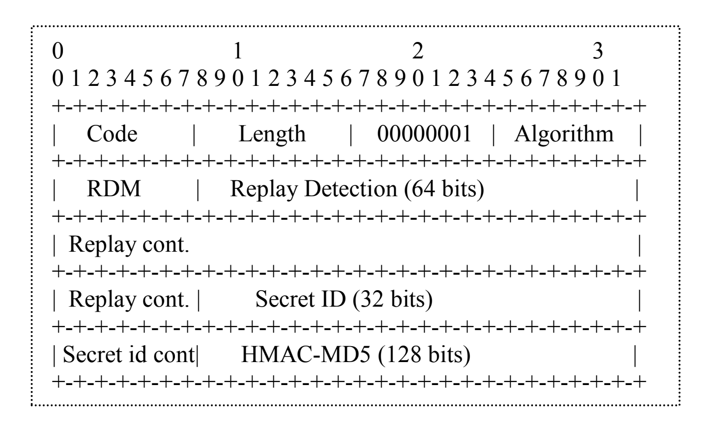


Figure 5-10 Format de l'option *Delayed authentication* dans les messages *DHCPOFFER*, *DHCPREQUEST*, *DHCPACK*

5.6.2.2.2 Bilan de ce mécanisme :

Les principaux problèmes de ce mécanisme sont la distribution et la flexibilité des clefs symétriques. Aucune de celles-ci n'affecte la sécurité du protocole, mais toutes les deux ont le potentiel d'affecter l'applicabilité d'employer réellement le protocole dans la pratique.

Ce protocole se fonde sur des clefs cryptographiques "déjà partagées" et connues par le client et par le serveur. L'un des principaux inconvénients d'usage de ces clefs est leur distribution compliquée. [RFC3118] précise que ce mécanisme a besoin d'un vrai mécanisme de signature numérique basé sur les algorithmes de chiffrement asymétrique tel que *RSA* [RFC3447] qui fournirait une meilleure sécurité. L'autre inconvénient (la flexibilité) de l'usage des clefs partagées devient évident quand le client change de réseaux. La méthode "*Delayed Authentication*" ne supporte pas l'authentification inter-domaine [CBDA]. Par conséquent, elle n'est pas bien dimensionnée dans une architecture où un client *DHCP* se connecte à de multiples domaines administratifs (un domaine administratif est un ensemble d'entités "client ou équipement" régissant sous le contrôle d'un même administrateur *DHCP*). En plus, elle est vulnérable à une attaque de déni de service par la saturation du serveur *DHCP* avec des messages *DHCPDISCOVER* qui ne sont pas authentifiés par ce protocole. Une telle attaque peut surcharger l'ordinateur sur lequel le serveur *DHCP* s'exécute et peut épuiser les adresses disponibles pour allocation.

Malgrès la robustesse de cette méthode d'authentification, nous n'avons pas trouvé de serveurs et de clients conformes à cet RFC [RFC3118]. Ensuite, aucune indication n'est fournie pour la gestion des clefs. Seule une personne, Ted Lemon de l'ISC (*Internet Software Consortium*) [ISC] a émis des recommandations pour cette gestion (le serveur *DHCP* de l'ISC intègre la majeur partie des distributions Linux ou BSD), en intégrant cette clef dans le fichier de configuration du client et du serveur, bien que le RFC définisse une technique pour la gestion des clefs (Annexe A) qui ressemble à une *IGC*.

Concernant ce mécanisme "*Delayed Authentication*", nous nous demandons pourquoi le premier message émis (*DHCPDISCOVER*) n'est pas authentifié, car le client en a la possibilité. En effet, une attaque visant à épuiser toutes les adresses IP reste possible car le serveur réserve tout de même une adresse après avoir reçu ce message. Le fait d'authentifier ce message n'alourdit pas la charge sur le réseau, et le serveur n'a qu'à vérifier la validité du *MAC* d'un message en plus sur toute la transaction, lui permettant d'authentifier le client avant de lui envoyer les informations de configuration.

Une autre critique viendra du fait que le serveur *DHCP* renvoie toujours en diffusion les informations de configurations en clair, et qu'aucune mesure n'ait été prise pour assurer la confidentialité de la suite de la transaction. On aurait pu imaginer que le serveur *DHCP* chiffrait ces informations avec la clef partagée, mettant les champs de l'en-tête à *NULL*, ainsi un client malicieux désirant se connecter sur le réseau n'aura aucune information. En effet, il est possible de déterminer une partie de la topologie du réseau, rien qu'en l'écoutant et qu'en observant les trames *DHCPOFFER* émises en diffusion, même sur un réseau local. Il est ainsi toujours possible à un client malicieux de savoir les adresses IP des machines les plus sensibles, ainsi que leurs fonctions. Une telle découverte faite de façon passive sur le réseau n'est pas acceptable.

5.6.3 Authentification *DHCP* via Kerberos V

Le mécanisme présenté ici (voir figure 5-11) est tiré d'un draft expiré [HOMDraft] mais l'originalité de cette proposition nous a poussé à la mentionner dans le présent rapport.

Dans cette méthode (Authentification *DHCP* via Kerberos V) [HOMDraft], les clients et serveurs *DHCP* utilisent une clef de session Kerberos [RFC1510] pour vérifier l'intégrité des messages échangés. Les données servant à la vérification sont incluses dans une nouvelle option des messages *DHCP*.

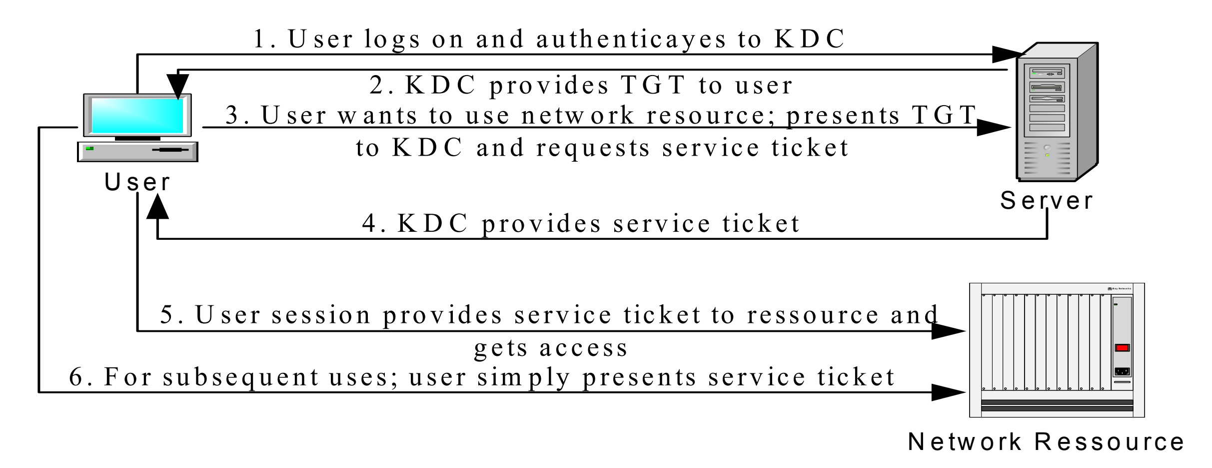


Figure 5-11 Schéma classique d'authentification Kerberos

5.6.3.1 Fonctionnement de l'authentification DHCP via Kerberos V

Voici une vue globale de ce mécanisme, tel qu'il a été présenté :

Le client *DHCP* soumet une requête de service "*AS_REQ*" incomplète au serveur *DHCP* dans le datagramme *DHCPDISCOVER*. Le serveur va ensuite compléter la requête *AS_REQ* en utilisant l'adresse IP qu'il souhaite proposer au client et l'envoyer au *KDC* (*Key Distribution Center*) faisant autorité chez le client dans le but d'obtenir un *TGT* (*Ticket Granting Ticket*) valide pour le client. Lorsque le *KDC* répond avec un message *AS_REP*, le serveur *DHCP* va extraire le *TGT* du client et va le soumettre au *KDC* avec son propre *TGT* dans le but d'obtenir un ticket "*user-to-user*" (utilisateur à utilisateur) pour dialoguer avec le client. La réponse *AS_REP* ainsi que la réponse *AP_REP* sont incluses par le serveur dans le message *DHCPOFFER*. Le client *DHCP* peut ainsi déchiffrer la partie *AS_REP* pour obtenir un *TGT* du royaume de son *KDC* ainsi qu'une clef qui lui servira à déchiffrer le ticket "*user-to-user*", extrayant ainsi une dernière clef connue seulement par lui et par le serveur. C'est celle-ci qui servira à authentifier le client et à vérifier l'intégrité de tous les messages échangés dans les messages *DHCPREQUEST* et *DHCPDECLINE*. Le serveur se servira de cette clef pour s'authentifier auprès du client dans les messages *DHCPOFFER*, *DHCPACK* et *DHCPNACK*.

5.6.3.2 Bilan de ce mécanisme

Dans l'authentification via le serveur Kerberos V, le client *DHCP* authentifiera, signera et protègera contre le rejeu les messages *DHCPREQUEST*, *DHCPDECLINE* et *DHCPRELEASE* en utilisant la clef de session "*user-to-user*" obtenue auprès du serveur *DHCP* et du *KDC*. Tant que le client n'a pas obtenu de clef de session, les messages *DHCPDISCOVER* ne pourront pas être authentifiés, sauf en utilisant des informations de pré-authentification que le serveur pourra vérifier auprès du *KDC*. Mais ce mécanime est fait de telle manière que l'intégrité du message ne sera plus garantie. De plus, cela ne fait que repousser le problème.

Un serveur *DHCP* pirate peut difficilement se faire passer pour un vrai s'il ne connaît pas le secret partagé entre le client et le *KDC*, en effet, le serveur fait parvenir au client un message qu'il ne peut décrypter qu'en connaissant ce secret. Cependant, il est possible que le serveur puisse extraire le *TGT* réservé au client ou même en faire la demande sans connaître le secret du client, ce qui n'est pas conforme à l’esprit dans lequel Kerberos a été créé. En effet, en cas de compromission du serveur,

n'importe qui peut se faire passer pour n'importe quel client et bénéficier de ses droits d'accès. Ceci est d'autant plus grave si Kerberos est utilisé pour contrôler l'accès à des services sensibles.

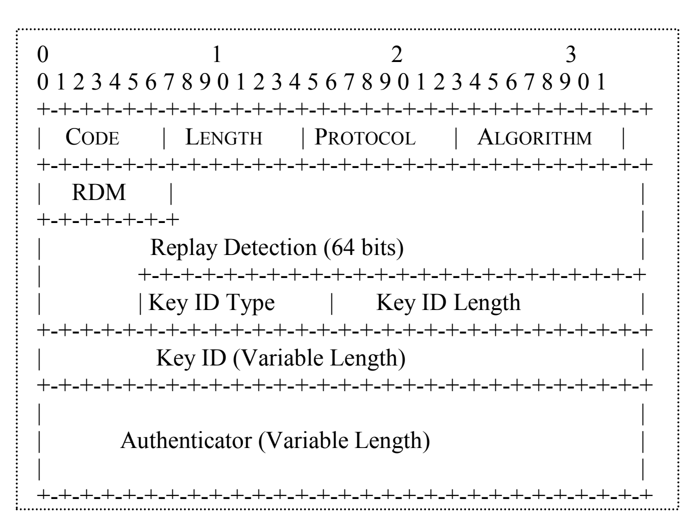
Cette méthode d'authentification par l'intermédiaire du serveur Kerberos V authentifie seulement le client et exige la communication avec ce serveur Kerberos en plus de la communication avec le serveur *DHCP*.

La complexité de ce mécanisme, ainsi que l'implémentation proposée n'ont pas permis au serveur Kerberos de s'imposer pour sécuriser *DHCP*, ce qui explique peut-être le rapide abandon des travaux qui allaient dans ce sens à l'*IETF*.

5.6.4 Authentification *DHCP* basé sue les Certificats (*CBDA*)

Une idée séduisante serait l'utilisation des certificats X.509 pour authentifier les machines désirant une adresse IP. [CBDA] propose une solution pour l'authentification par certificat d'identité X.509 dans *DHCP*. L'idée principale est de proposer une méthode de gestion des clefs un peu plus élaborée que celle proposée dans [RFC3118] et de garder un mécanisme d'authentification au moins aussi sûr que le mécanisme *"Delayed Authentication*".

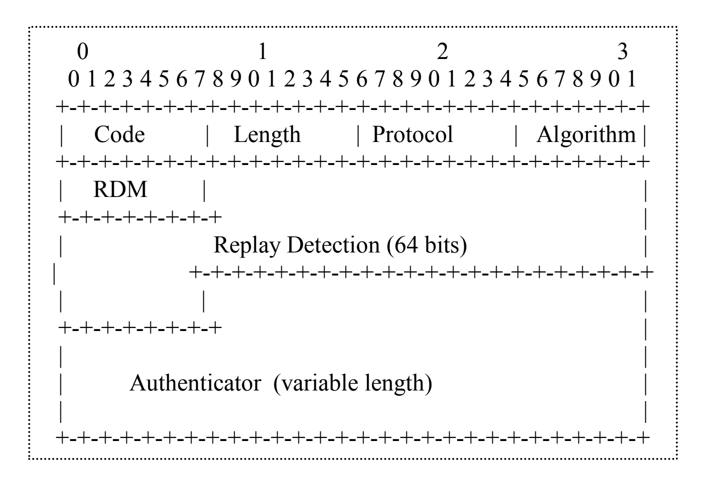
D'ailleurs, les messages échangés ressemblent beaucoup au mécanisme précédemment cité, sauf que le client va joindre son certificat dans le message *DHCPDISCOVER* et le serveur *DHCP* joindra le sien dans le message *DHCPOFFER*. Le format général de l'option pour ces deux messages est le suivant:



Avec respectivement :

- *Code* : Le code de l'option (0x5A) pour l'authentification.
- *Length* : La longueur de l'option.
- *Protocol* : Le code du protocole utilisé (2 pour *CBDA*).
- *Algorithm* : L'algorithme utilisé pour la signature ou le chiffrement(*RSA-MAC* ou *DSA*).
- *RDM* : La méthode de détection des tentatives de rejeu (*Replay Detection Method*).
- *Replay Detection* : L'information pour le champ précédent.
- *Key ID Type* : Le type de clef utilisée (ici 0 pour les certificats X.509).
- *Key ID Length* : La longueur de cette clef.
- *Key ID* : La clef elle-même (ici le certificat).
- *Authenticator* : La signature ou le chiffré du paquet entier (sauf des champs *"hops"* et *"giaddr"*).

Ensuite les deux entités vont utiliser les informations contenues dans les certificats pour le reste de la transaction. Le format de l'option devient le suivant:



Avec respectivement :

- *Code* : Le code de l'option (0x5A) pour l'authentification.
- *Length* : La longueur de l'option.
- *Protocol* : Le code du protocole utilisé (2 pour *CBDA*).
- *Algorithm* : L'algorithme utilisé pour la signature ou le chiffrement(*RSA-MAC* ou *DSA*).
- *RDM* : La méthode de détection des tentatives de rejeu (*Replay Detection Method*).
- *Replay Detection* : L'information pour le champ précédent.
- *Authenticator* : La signature ou le chiffré du paquet entier (sauf des champs *"hops"* et *"giaddr"*).

Le serveur et le client *DHCP* gardent en mémoire, durant au moins toute la durée du bail, leur certificat respectif. De cette façon, les certificats n'ont pas besoin d'être ré-émis, mais il est nécessaire que le serveur conserve une table de correspondance. Le reste de la transaction est identique à la méthode *Delayed Authentication*. Il est à noter que cette méthode reste sensible à une attaque de type "*man-in-the-middle*", mais les auteurs de [CBDA] suggèrent que le client connaisse à l'avance les autorités de certification qui connaissent le certificat du serveur.

Cette solution semble rapidement déployable tout en apportant une sécurité supplémentaire que celle proposée dans [RFC3118], c'est-à-dire que l'attaque visant à épuiser les adresses proposées par un serveur *DHCP* ne semble plus possible. De plus, cette solution étant basée sur les certificats, l'utilisation d'une *IGC* standard est envisageable.

Cette méthode d'authentification utilise les certificats d'identité X.509 pour authentifier les entités *DHCP*. [CBDA] implique l'envoie du certificat d'identité X.509 ou la chaîne de certification avec un signataire commun, entre le client et serveur *DHCP*, comme option dans les paquets *DHCPDISCOVER* et de *DHCPOFFER*, et puis celui seulement des paquets signés dans les paquets *DHCPREQUEST* et *DHCPACK*.

5.6.4.1 Bilan de ce mécanisme:

La taille maximum standard (576 bytes) d'un message *DHCP* peut être trop courte pour contenir les certificats d'identité X.509 ou les chaînes de certification. Les certificats d'identité X.509 sont très grands et les paquets *DHCP* ont été conçus à l'origine pour être relativement petits. Ainsi, les clients implémentant le mécanisme *CBDA* devraient envoyer l'option "*Maximum DHCP Message Size*" [RFC2132] si la pile *TCP/IP* du client *DHCP* est capable de recevoir de plus grands datagrammes IP. Dans ce cas, le client devrait placer la valeur de cette option au moins à la valeur de *MTU* (*Maximum*

Transmission Unit) de l'interface qu'il configure. Il peut placer la valeur de cette option plus haute, jusqu'à la taille du plus grand paquet *UDP* qu'il est préparé à accepter. Notons que la valeur spécifiée dans l'option "*Maximum DHCP Message Size*" est égale à la taille maximum du paquet totale, y compris des en-têtes IP et *UDP*. Les clients *DHCP* demandant cette option et les serveurs *DHCP* envoyant cette option doivent implémenter l'option "*DHCP concatenation*" [RFC3396]. Dans [RFC2131], aucune limite universelle n'existe pour la taille des paquets *DHCP*, mais dans la pratique, il y a une nécessité de mettre une limite à la taille des paquets *DHCP* pour empêcher la saturation d'un hôte. Les clients peuvent spécifier la longueur maximum du paquet *DHCP* qu'ils accepteront, et plusieurs de ces limites peuvent devoir être redisposées si de longues chaînes de certificats sont employées.

5.7 Conclusion

Le protocole *DHCP* est un standard de l'*IETF* basé sur le protocole *BOOTP*, dont il pallie les limites, en permettant l'allocation dynamique et centralisée des adresses IP à toutes les machines d'un réseau TCP/IP. Destiné à faciliter le travail des administrateurs systèmes en automatisant l'attribution des adresses IP aux clients du réseau, le protocole *DHCP* souffre de nombreux problèmes de sécurité. Le protocole *DHCP* ne supporte pas le mécanisme avec lequel les clients et les serveurs *DHCP* s'authentifient. De plus, le protocole *DHCP* n'assure pas l'intégrité des données échangées, ni leur confidentialité. Quelques solutions [RFC3118][HOMDraft][CBDA] ont tenté de renforcer la sécurité du protocole *DHCP* afin d'assurer l'authentification d'entité (client et/ou serveur) *DHCP* et/ou des contenus des messages *DHCP*.

Dans ce chapitre nous avons exposé le protocole *DHCP*, son fonctionnement, ces limites, les différentes solutions qui ont tenté de renforcer sa sécurité, ainsi que les limites de ces solutions.

Afin de permettre un contrôle strict sur les équipements par authentification forte, nous présenterons dans le chapitre suivant une extension du protocole *DHCP* à savoir *E-DHCP* qui remède à ces problèmes en assurant l'authentification d'entité (client et serveur) *DHCP* et des contenus des messages *DHCP*.

Chapitre 6

6 *E-DHCP* (Extended-Dynamic Host Configuration Protocol)

Résumé

Afin de permettre un contrôle strict sur les équipements par authentification forte, nous présentons dans ce chapitre une extension du protocole DHCP à savoir E-DHCP (Extended Dynamic Host Configuration Protocol). Cette extension de DHCP est possible par l'ouverture des options prévues de ce protocole. Cette extension consiste à rajouter une nouvelle option DHCP qui fournit simultanément l'authentification d'entités et de messages DHCP, en se basant sur l'utilisation de clefs de chiffrement asymétrique RSA, de certificats d'identité X.509 et de certificats d'attributs.

De plus, nous attribuons à un nouveau serveur E-DHCP les fonctionnalités du serveur AA (Attribute Authority) d'un PMI (Privilege Management Infrastructure). E-DHCP crée un certificat d'attributs pour le client contenant l'adresse Internet attribuée dynamiquement. Pour leur authentification au sein des architectures réseaux, les équipements peuvent faire preuve de l'authenticité de leur adresse en présentant leur certificat d'identification X.509 et le certificat d'attributs.

6.1 Concepts de base de *E-DHCP*

En raison des vulnérabilités des mécanismes d'authentification des entités et des messages *DHCP* décrites dans le chapitre précédent, il s'avère nécessaire de trouver un mécanisme d'authentification plus robuste.

Dans ce chapitre, nous proposons une extension du protocole *DHCP,* appelée *E-DHCP (Extended Dynamic Host Configuration Protocol)* [DEMERDraft]. Cette extension assure l'authentification de l'entité (client et serveur) et du message *DHCP* d'une part, et le contrôle d'accès dans le système *DHCP* d'autre part.

6.1.1 Principes

L'extension *E-DHCP* [ICETE] [SEC] est basée sur les principes suivants:

1. Définition d'une nouvelle option *DHCP* [RFC2489] qui fournit simultanément l'authentification d'entités et l'authentification des messages *DHCP*. La technique utilisée par cette option est basée principalement sur l'utilisation de clefs de chiffrement asymétrique *RSA*, de certificats d'identité X.509 et de certificats d'attributs.
2. Attribution à un nouveau serveur (*E-DHCP*) des fonctionnalités du serveur *AA (Attribute Authority)* [RFC3281] d'un *PMI (Privilege Management Infrastructure)* [RFC3281]. Le serveur *E-DHCP* regroupe un serveur *DHCP* et un serveur *AA* (voir figure 6-1). Ce serveur crée un certificat d'attributs pour le client lors de l'envoi du message *DHCPACK* afin d'assurer la relation entre le certificat d'identité du client et l'adresse IP allouée. Ce certificat d'attributs sera ensuite utilisé dans le contrôle d'accès aux services autorisés.

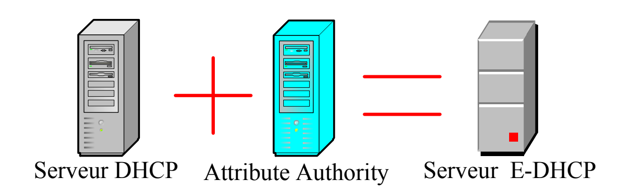


Figure 6-1 Serveur *E-DHCP*

L'authentification et le contrôle d'accès, qui sont les deux objectifs de cette proposition sont illustrés dans le schéma suivant (voir figure 6-2) qui présente l'architecture générale de *E-DHCP*.

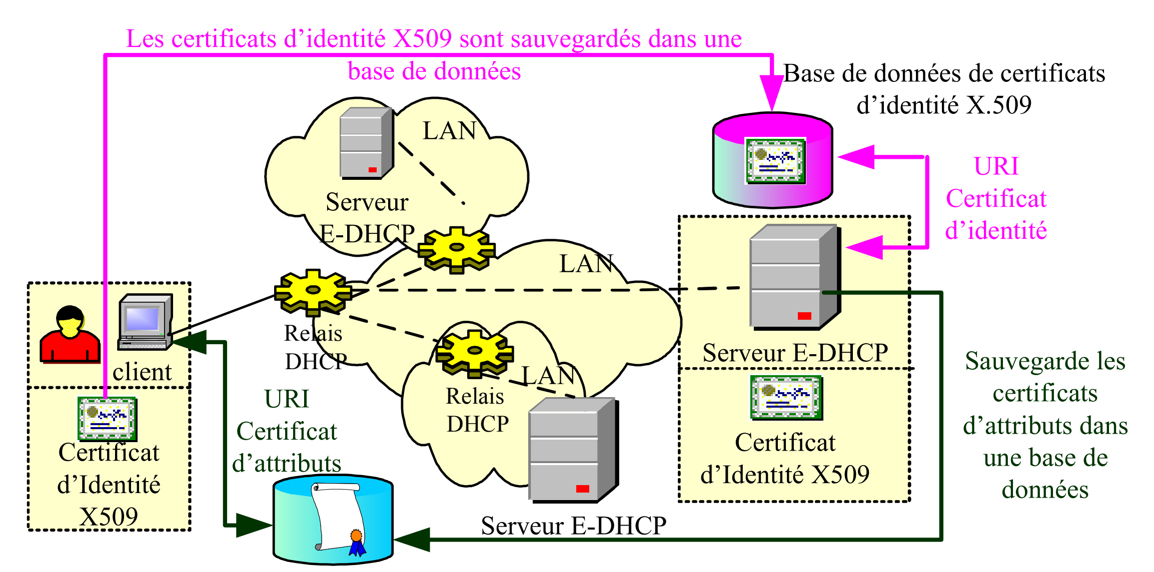


Figure 6-2 Architecture générale de *E-DHCP*

Les différents composants de l'architecture *E-DHCP* sont présentés dans la figure 6-3:

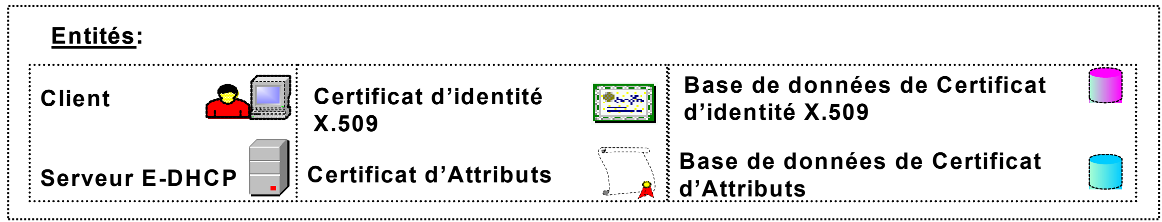


Figure 6-3 Les entités de l'architecture *E-DHCP*

1. ***Client***: Un "client" ou un "client *E-DHCP*" est un équipement qui utilise *E-DHCP* pour obtenir des paramètres de configuration.
2. ***Serveur E-DHCP***: C'est un serveur qui:
   a) renvoie l'adresse IP et les paramètres de configuration aux clients *E-DHCP* suite à leur demande.
   b) crée un certificat d'attributs au client contenant l'adresse IP allouée.
3. ***Base de données des certificats d'identité X.509***: C'est une base de données où les certificats d'identité X.509 des entités (client ou serveur) sont sauvegardés.

4. ***Base de données des certificats d'attributs***: C'est une base de données où les certificats d'attributs des clients sont sauvegardés.

Les détails concernant l'interaction entre ces composants seront présentés dans la figure 6-7.

6.1.2 Conditions nécessaires de *E-DHCP*

La réalisation de cette extension requiert quelques conditions préalables :

1. Le client *E-DHCP* doit détenir un certificat d'identité X.509 valide délivré par une autorité de confiance (*CA*).
2. Le serveur *E-DHCP* doit avoir un certificat d'identité X.509 valide délivré par une autorité de confiance (*CA*).
3. Chaque entité (client et serveur *E-DHCP*) doit pouvoir valider le certificat de l'autre.

6.2 Solution *E-DHCP* : mode d'emploi

6.2.1 Principes

L'extension proposée vise à fournir une authentification forte au sein de *DHCP*. Les deux principes mis en œuvre sont :

1. L'authentification mutuelle de l'usager du réseau et du serveur *DHCP*. Cette authentification mutuelle va permettre à un client *DHCP* de se connecter à un serveur *DHCP* authentique, lequel pourra également s'assurer de l'identité du client.
2. La mise en place d'un contrôle d'accès du client à des services réseau basés sur cette authentification.

6.2.1.1 Ajout d'une option d'authentification

L'ajout de nouvelles options dans une trame *DHCP* a été prévu dès le départ dans le protocole *DHCP* [RFC2131]. Un RFC «*Procedure for Defining New DHCP Options*» [RFC2489] dédié traite d'ailleurs le sujet. Les paramètres de configuration et autres informations de contrôle sont transportés dans les champ "*options*" du message *DHCP*. Le format général d'une trame *DHCP* est le suivant:

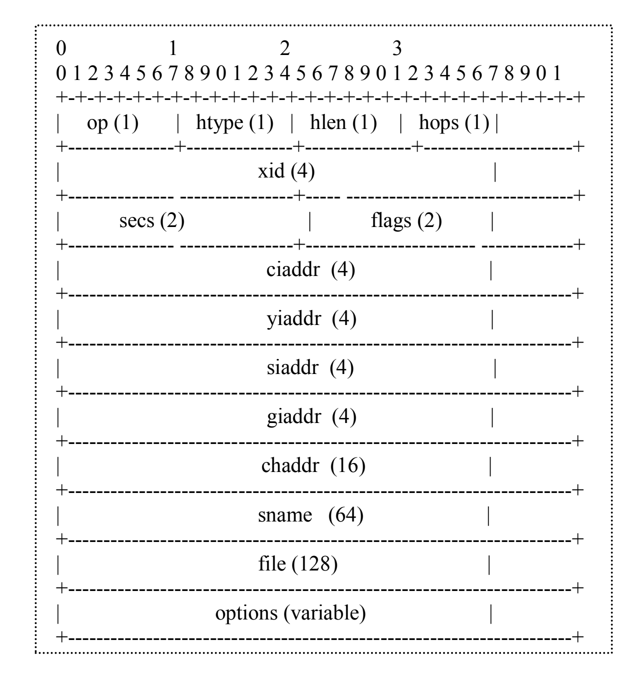


Figure 6-4 Format d'une trame *DHCP* – figure tirée du [RFC2131]

On constate donc qu'un champ "*options*" de longueur variable est prévu pour permettre de définir, suivant les besoins de constructeur, des nouveaux options. Un client DHCP doit être préparé pour recevoir des messages *DHCP* avec un champ "options" d'au moins 312 octets. Ce qui implique qu'un client *DHCP* doit être préparé pour la réception d'un message jusqu'à 576 octets, taille minimum d'un datagramme qu'une machine IP doit être préparée à accepter [RFC1122]. Les clients *DHCP* doivent négocier l'utilisation de messages *DHCP* plus grands à l'aide de l'option "*taille maximum message DHCP*". Le champ d'options peut être étendu d'avantage dans les champs "*file*" et "*sname*".

Le diagramme suivant (voir figure 6-5) définit le format de l'option d'authentification du *DHCP* proposé [DEMERDraft]. Ce format sera inséré dans le champ "*paramètres complémentaires*" (voir figure 5-1, chapitre cinq) du paquet *DHCP*.

Les champs de cette option d'authentification proposée de l'architecture de *E-DHCP* sont décrits comme suit :

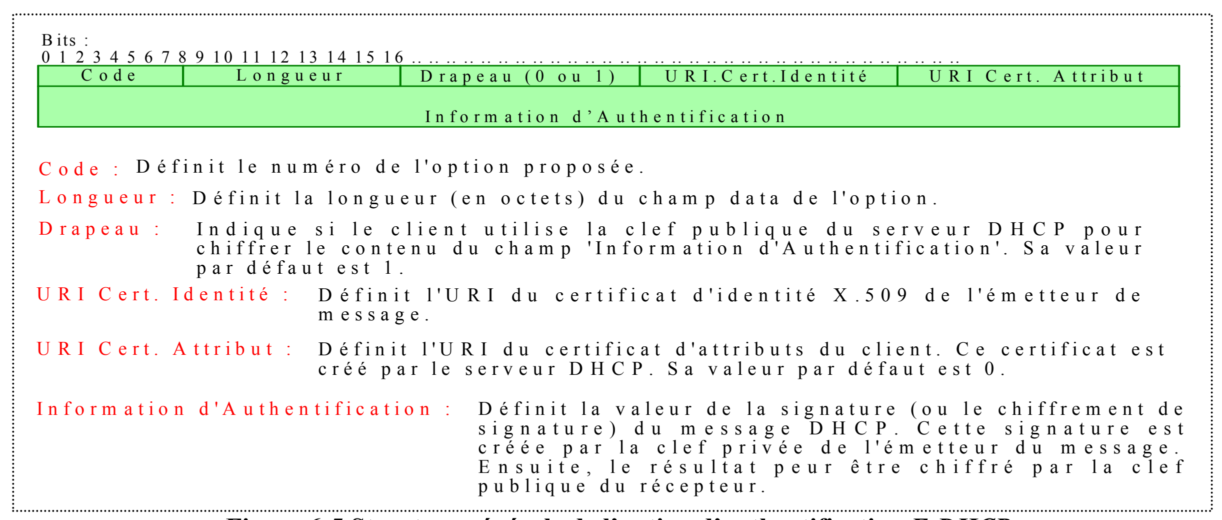


Figure 6-5 Structure générale de l'option d'authentification *E-DHCP*

Le "*Code*" est propre à chaque option et identifie donc cette nouvelle option. Le champ "*longueur*", comme on peut s'y attendre, donne la taille totale du bloc de données propre à cette option. Le champ "*Drapeau*" vaut 1 si l'émetteur du paquet a utilisé la clef publique du destinataire pour chiffrer le champ "*Information d'Authentication*", sinon 0. Le champ "*Drapeau*" est mis à 1 dans tous les messages échangés, sauf dans un message *DHCPDISCOVER* si le client ne connaît pas la clef publique du serveur *E-DHCP*.

Le champ "*URICert.Identité*" précise l'URI [RFC2396] du certificat d'identité X.509 associé à l'émetteur du message (nous verrons par la suite comment les certificats sont gérés et stockés). Le champ "*URICert.Attribut*" indique l'URI du certificat d'attributs propre au client. Enfin, le champ "*Information d'Authentication*" est une signature calculée par l'émetteur au moyen de sa clef privée, éventuellement chiffrée avec la clef publique du destinataire dans le cas où le champ "*Drapeau*" vaut 1.

On remarque ainsi que les messages échangés porteront des signatures qui garantiront à chaque partie l'authenticité de l'identité de l'autre partie, puisque chaque message sera signé par son émetteur.

6.2.1.2 Capacités d'identification et d'autorisation

La solution *E-DHCP* [SAR04] [ICTTA] repose principalement sur la notion de certificats. Il s'avère nécessaire de rappeler brièvement quelques notions utiles sur ces certificats avant d'aller plus loin (pour plus de détails concernant les certificats, veuillez voir les chapitres 2 et 3).

Un certificat est un document établissant un lien entre une clef publique et un sujet ou un privilège. Cette liaison dépend du contexte et du type du certificat.

Les certificats d'identité X.509 ont pour objet d'associer à chaque utilisateur, dont l'identité est précisée dans le certificat, sa clef publique. X.509 est le format de certificats à clef publique le plus utilisé.

Les certificats d'attributs X.509, quant à eux, ont pour objet d'associer à chaque utilisateur la liste de ses autorisations ou privilèges. Ces deux types de certificats seront signés par des tiers de confiance, ce qui permettra de les authentifier facilement. Les certificats d'attributs peuvent supporter et mettre en application une partie significative du processus d'autorisation. L'idée fondamentale est que toutes les décisions de contrôle d'accès ne sont pas basées sur l'identité. Les décisions de contrôle d'accès basées sur le rôle, les règles et les rangs exigent des informations supplémentaires. Du moment qu'un certificat d'attributs ne contienne pas de clef publique, il doit être utilisé conjointement avec un certificat d'identité, tel qu'un certificat X.509.

Ces deux types de certificats sont signés par des tiers de confiance, ce qui permettra de les authentifier facilement. Le certificat racine utilisé par le tiers de confiance signant les certificats d'identité (*Certificates Authority*) devra être connu du serveur *E-DHCP*. Pour la signature des certificats d'attributs, le serveur *E-DHCP* jouera lui-même le rôle de tiers de confiance (*Attributes Authority*).

A côté de la *IGC* (*Infrastructure de Gestion des clefs*) qui offre la gestion des certificats d'identité, il convient donc de mettre en place une IGP (*Infrastructure de Gestion des privilèges).* Cette *IGP* sera administrée par le serveur *E-DHCP*.

6.2.1.3 Fonctionnement du DHCP classique

On rappelle très brièvement le fonctionnement classique de *DHCP*. Un client qui cherche à configurer ses paramètres réseaux et notamment à obtenir une adresse IP commence par envoyer en *broadcast* un message *DHCPDISCOVER*. Ce premier message constitue une demande de proposition de la part du client. Celui-ci souhaite que les serveurs *DHCP*, éventuellement à l'écoute, lui envoient les offres qu'ils peuvent lui faire. Les serveurs qui ont une proposition à faire envoient alors un message *DHCPOFFER* décrivant ces offres et précisant l'adresse de réponse. Le client en choisit une

puis indique son choix en envoyant un paquet *DHCPREQUEST*. Enfin, le serveur confirme que l'attribution a été effectuée par un message *DHCPACK*.

Ce fonctionnement est illustré par la figure ci-dessous (voir chapitre 5, page 79, figure 5-3).

6.2.1.4 Fonctionnement de E-DHCP

Les messages échangés dans *E-DHCP* sont exactement les mêmes que ceux dans *DHCP* et dans le même ordre (voir figure 6-6).

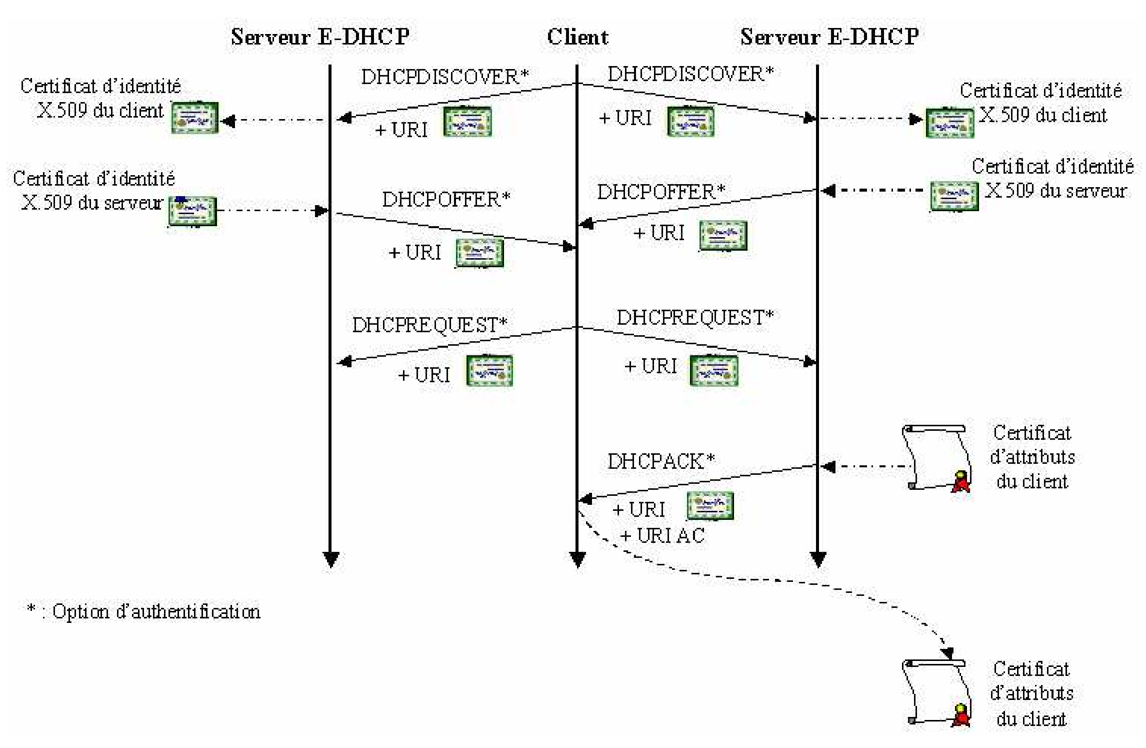


Figure 6-6 Scénario de configuration avec *E-DHCP*

Toutefois, on ajoute l'option d'authentification à chacun d'entre eux. Ainsi, lors du *DHCPDISCOVER*, le client ajoute cette option et précise l'*URI* qui permettra de récupérer son certificat d'identité auprès de la *IGC*. La valeur du champ donnant l'*URI* du certificat d'attributs est nulle, puisque aucune attribution d'adresse IP n'a été faite pour l'instant. Enfin, la signature électronique du client est réalisée au moyen de sa clef privée. Cette signature est appliquée sur tout les champs du message *DHCP* (y compris l'en-tête et les options). Cette signature pourra être vérifiée par les serveurs *E-DHCP* puisque qu'ils ont accès au certificat d'identité du client et donc à sa clef publiquet.

Si le client ne connaît pas la clef publique du serveur *E-DHCP*, la valeur du champ "*Drapeau*" sera 0 et la valeur du champ "*Information d'Authentication*" sera la valeur de la signature.

En revanche, si le client connaît la clef publique du serveur *E-DHCP*, la signature sera chiffrée par la clef publique du serveur *E-DHCP*. Dans ce cas, la valeur du champ "*Drapeau"* sera 1 et la valeur du champ "*Information d'Authentication*" sera la valeur de la signature chiffrée.

Enfin, le client encapsule l'option d'authentification dans un message *DHCPDISCOVER* (voir figure 6-7), puis l'envoie au serveur *E-DHCP*.

Bits :
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 .. ..

Code	Longueur	Drapeau (0 ou 1)	URI.Cert.Identité = *www.enst.fr/Database1/ CertificatIdentitéClient1*	URI Cert. Attribut = 0
Information d'Authentification = *Valeur de la Signature* ou *Valeur de la 'Signature chiffrer'*				

Figure 6-7 Structure de l'option d'authentification dans les messages *DHCPDISCOVER* et *DHCPINFORM*

Le serveur *E-DHCP*, en recevant le message *DHCPDISCOVER*, utilise l'*URI* du certificat d'identité du client contenu dans le message pour extraire la clef publique du client. Le serveur vérifie la valeur contenue dans le champ "*Drapeau*", puis vérifie l'authentification du client et du message *DHCPDISCOVER*.

Si la valeur de ce champ ("*Drapeau*"*)* est égale à 1, le client connaît donc la clef publique du serveur et il l'a utilisée pour chiffrer la signature. Dans ce cas, le serveur utilise sa clef privée pour déchiffrer la valeur contenue dans le champ "*Information d'Authentication*", ensuite le résultat de ce déchiffrement (valeur de la signature) sera déchiffré par la clef publique du client.

Si la valeur de ce champ (*"Drapeau")* est égale à 0, le client ne connaît donc pas la clef publique du serveur. Dans ce cas, le serveur utilise la clef publique du client pour déchiffrer la valeur contenue dans le champ "*Information d'Authentication*".

Si un serveur recevant le message *DHCPDISCOVER* a bien authentifié le client, il peut préparer et envoyer une offre au client en envoyant un message *DHCPOFFER* (voir figure 6-8). Cette fois, l'option contiendra l'URI du certificat d'identité du serveur, un champ nul pour l'URI du certificat d'attributs (aucune adresse IP n'a été attribuée pour l'instant), puis une signature réalisée grâce à la clef privée du serveur, et chiffrée au moyen de la clef publique du client, qui est désormais connue du serveur (le "*Drapeau*" est donc à 1).

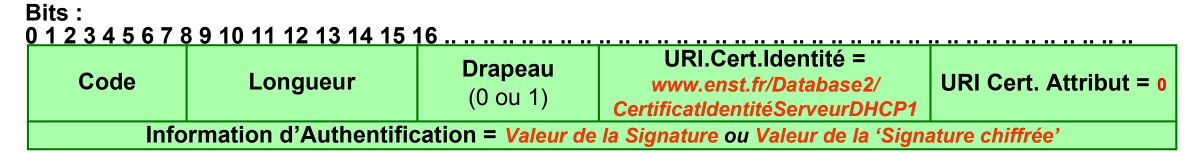
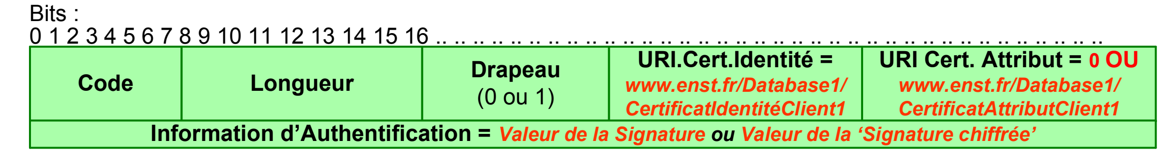


Figure 6-8 Structure de l'option d'authentification dans le message *DHCPOFFER*

Dans l'hypothèse où le client a bien authentifié le serveur *E-DHCP* lors de l'étape précédente, et si l'offre qui lui est faite lui convient, il va répondre par un message *DHCPREQUEST* (voir figure 6-9). La valeur du champ "*URICert.Identité*" est la même que pour le message *DHCPDISCOVER*. Le "*Drapeau*" vaut 1 et la signature est calculée comme précédemment, en chiffrant le résultat grâce à la clef publique du serveur. En revanche, l'URI du certificat d'attributs serait non nulle si le *DHCPDISCOVER* est effectué dans le cadre d'un renouvellement de bail. En effet, dans ce cas, le serveur *E-DHCP* a déjà procédé à la création du certificat d'attributs pour le client, et a donc joué son rôle de tiers de confiance. Dans le cas où il s'agit d'une première attribution, ce champ serait nul.


Bits :
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 .. ..

Code	Longueur	Drapeau (0 ou 1)	URI.Cert.Identité = *www.enst.fr/Database1/ CertificatIdentitéClient1*	URI Cert. Attribut = 0 OU *www.enst.fr/Database1/ CertificatAttributClient1*
Information d'Authentification = *Valeur de la Signature* ou *Valeur de la 'Signature chiffrée'*				

Figure 6-9 Structure de l'option d'authentification dans les messages *DHCPREQUEST*, *DHCPDECLINE* et *DHCPRELEASE*

Enfin, le serveur authentifie le message précédent. S'il ne détecte aucune anomalie, il envoie un message *DHCPACK* (pour confirmer au client l'offre de configuration qu'il a déjà proposée) (voir figure 6-10), avec l'option remplie comme précédemment. Dans ce cas, il crée un certificat d'attributs pour le client, le rend disponible dans la *IGP* (en le sauvagardant dans la base de données des certificats d'attributs), et lui indique, dans le champ "*option*", l'URI permettant d'y accéder. Le client récupère ce certificat et le conserve soigneusement. Si le serveur détecte une anomalie, il renvoit un message *DHCPNACK* (ce message permet à un serveur de refuser d'offrir une configuration).

Bits :
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 .. ..

Code	Longueur	Drapeau (0 ou 1)	URI.Cert.Identité = *www.enst.fr/Database2/ CertificatIdentitéServeurDHCP1*	URI Cert. Attribut = *www.enst.fr/Database1/ CertificatAttributClient1*
Information d'Authentification = *Valeur de la Signature* ou *Valeur de la 'Signature chiffrée'*				

Figure 6-10 Structure de l'option d'authentification dans le message *DHCPACK*

Dès l'achèvement de cette phase et l'authentification du message *DHCPACK* par le client, celui-ci peut se servir de l'adresse IP qui lui est affectée, de son certificat d'identité et de son certificat d'attributs pour s'authentifier auprès d'un serveur de contrôle d'accès. Ceci permet de filtrer l'accès à des services en fonction de l'identité et des attributs d'un utilisateur. Ce procédé est totalement générique et applicable à un vaste champ de services. Le contenu de certificats d'attributs sera présenté dans les paragraphes suivants.

Il est inutile de reproduire une nouvelle figure puisque le procédé est le même. Ce sont les mêmes messages qui sont échangés, mais ils sont complétés par l'option d'authentification.

6.2.1.5 Scénario E-DHCP

Avant de présenter un scénario usuel de la configuration d'un client par *E-DHCP* (voir figure 6-11) où le client possède au préalable le certificat X.509 du serveur (le cas le plus courant), présentons tout d'abord les notations utilisées :

- S : serveur *E-DHCP*
- C : client *E-DHCP*
- CI : certificat d'identité
- CA : certificat d'attributs
- P_X : clef publique associée au certificat X.509 de l'entité X
- S_X : clef privée associée à la clef P_X
- $Cert_X$: certificat X.509 de l'entité X
- $P_X(Z)$: chiffrement du message Z avec la clef publique P_X
- $S_X(Z)$: raccourci pour la signature du message Z par la clef privée S_X

- M_0 : message à envoyer, avant toute opération cryptographique (chiffrement, signature, …)
- M : message envoyé ou reçu
- G – {A, B} : le message G diminué des champs A et B

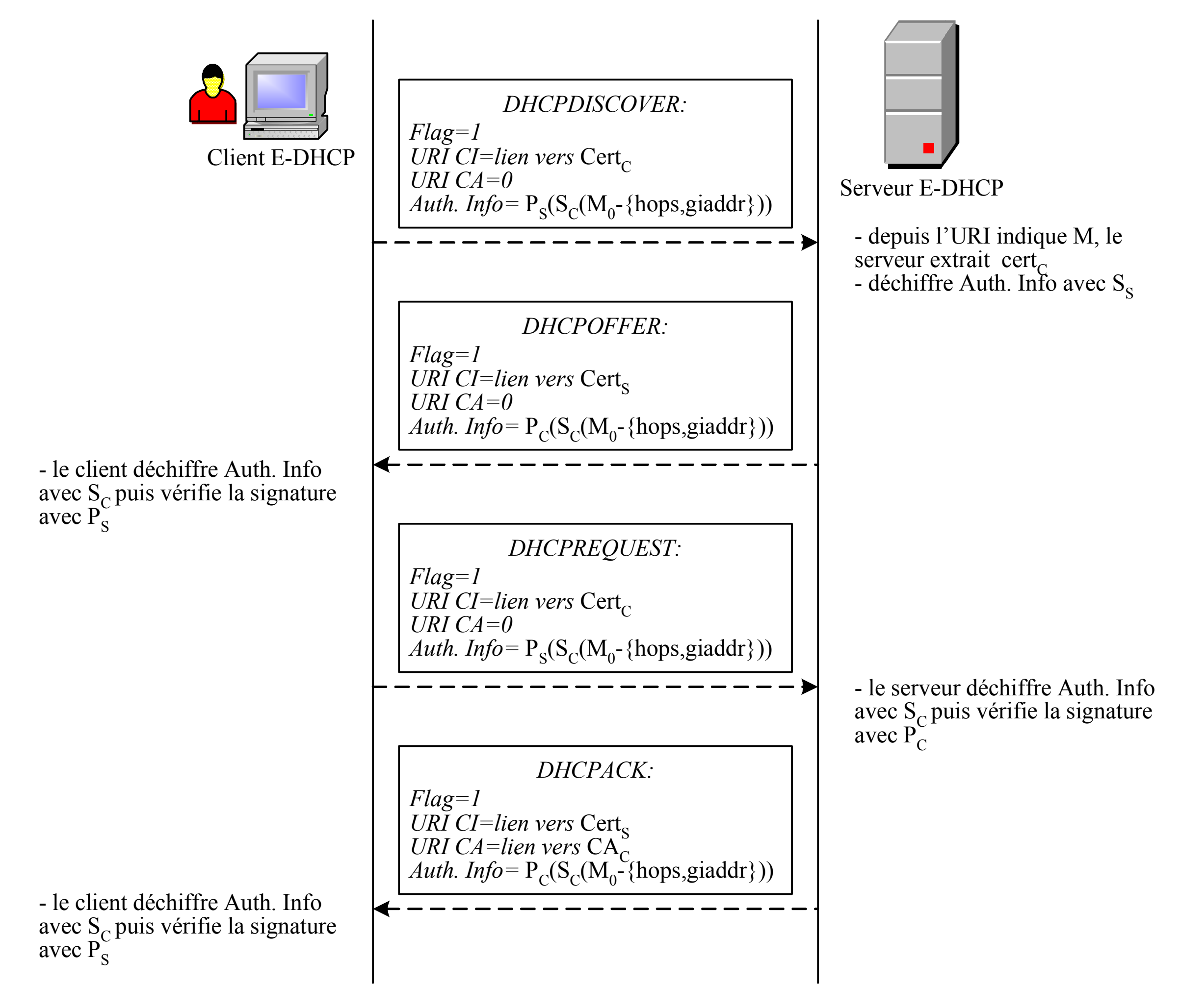


Figure 6-11 Scénario usuel de la configuration d'un client par *E-DHCP*

Une vue détaillée d'ensemble du scénario de système *E-DHCP* (côté client et serveur) est présentée dans l'annexe C. Le contenu des certificats d'attributs est présenté dans le paragraphe suivant.

6.2.1.6 Certificat d'attributs

La configuration du client est contrôlée par son certificat d'attributs lié à son certificat d'identité X.509 via le champ "*Holder*". Ce certificat d'attributs (voir figure 6-12) est associé, à la fois à l'identité du client, et au bail accordé pour l'utilisation d'une adresse IP.

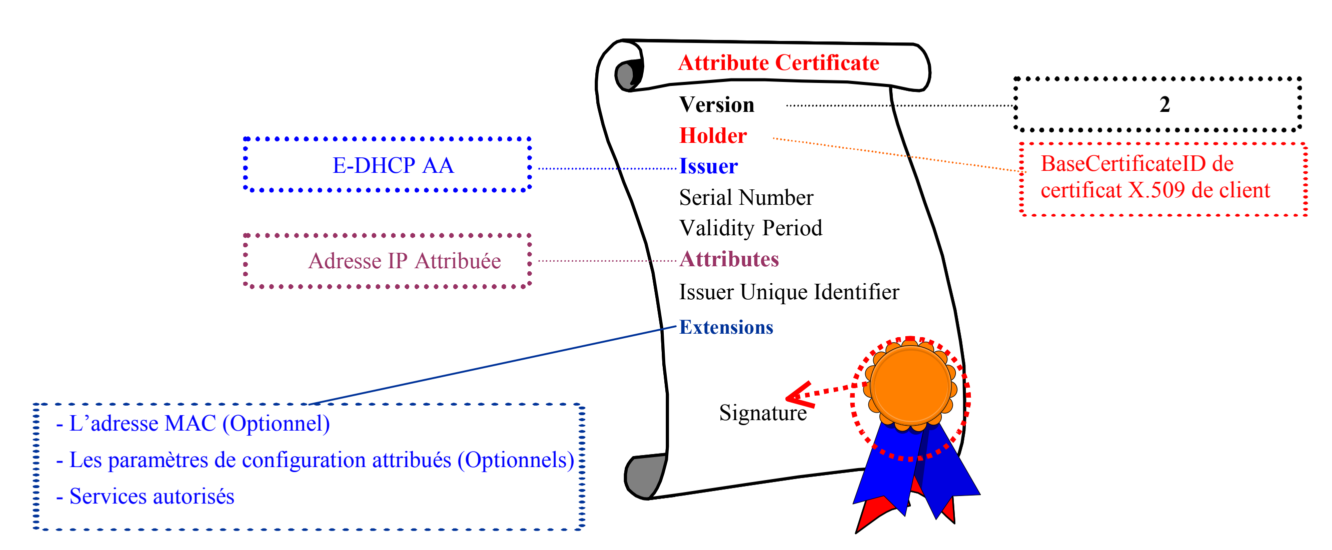


Figure 6-12 Contenu du certificat d'attributs créé par le serveur *E-DHCP*

Le certificat d'attributs généré par le serveur *E-DHCP* contient les champs suivants:

1. ***Version***: précise la version du certificat d'attributs. La valeur actuelle de ce champ est 2.
2. ***Holder***: décrit le détenteur du certificat d'attributs. Ce champ permet de relier ce certificat d'attributs à un certificat d'identité X.509 (voir figure 6-14). Dans notre cas, "*Holder=baseCertificateID*"**,** "*holder*" correspond donc à la concaténation des champs "*issuer*" et "*SerialNumber*" du certificat d'identité X.509 du client.
3. ***Issuer***: identifie l'entité (*E-DHCP*) qui a signé et émis le certificat d'attributs.
4. ***Signature***: contient l'identifiant de l'algorithme utilisé par le serveur *E-DHCP* pour signer les certificats d'attributs.
5. ***Serial Number***: contient un numéro de série qui identifie de façon unique le certificat d'attributs généré par le serveur *E-DHCP*.
6. ***Attr Cert Validity Period***: précise l'intervalle de validité du certificat d'attributs (qui correspondra à celui du bail). À partir du moment où le bail expire, le client ne peut plus dépasser le serveur de contrôle d'accès.
7. ***Attributes***: contient les attributs associés au détenteur du certificat d'attributs. Dans le cas étudié, l'adresse IP affectée au client par le serveur *E-DHCP* sera indiquée dans ce champ.
8. ***Issuer Unique ID***: est utilisé pour identifier l'émetteur du certificat d'attributs dans le cas où le champ "*issuer*" n'est pas suffisant.
9. ***Extensions***: permet de rajouter de nouveaux champs au certificat d'attributs, notamment:
   a) les paramètres de configuration affectés par le serveur *E-DHCP* (optionnels).
   b) l'adresse *MAC* de l'équipement (optionnelle). Le serveur *E-DHCP* peut allouer une adresse IP à un utilisateur identifié qui utilise une machine précise (possédant cette adresse *MAC*). Dans ce cas, on relie l'identité de l'utilisateur et l'adresse *MAC* à l'adresse IP.

6.2.2 Attaques possibles

Dans cette section, nous présentons quelques attaques possibles sur *E-DHCP* [ICETE] [SEC], ainsi que les précautions que nous avons prises pour remédier à ces attaques.

6.2.2.1 Rejeu

Si un attaquant récupère un message échangé entre un client et un serveur *E-DHCP*, par exemple un *DHCPDISCOVER* et le rejoue plus tard sans le modifier, alors le message sera considéré comme valide par le serveur, car la signature du message reste valable.

La [RFC2131] définit le champ "*xid*" comme étant l'identifiant de la transaction, initialisé à une valeur aléatoire par le client. Or, ce champ est optionnel, de plus, il peut être, selon les implémentations, incrémenté à chaque message ou bien à chaque session.

Ainsi, dans le cas de *E-DHCP*, il faut requérir la présence de ce champ et aussi son incrémentation à chaque échange pour éviter les attaques par rejeu. Certes, l'incrément à appliquer au champ "*xid*" est prévisible, mais l'intégrité et l'authenticité du message sont garanties grâce à la signature (chiffrée ou pas) présente dans le champ "*Information d'Authentication*" du champ options de *E-DHCP*.

6.2.2.2 Faux serveur E-DHCP sur le réseau :

Un faux (mal intentionné) serveur *E-DHCP* peut tromper un client, si celui-ci ne connaît pas son serveur *E-DHCP* [DEMERDraft]. Pour que cette attaque fonctionne, il faut que le serveur fasse partie de la même chaîne de certification que le client. Ainsi, la sécurité de *E-DHCP* repose entièrement sur la chaîne de certification. Si la chaîne de certification est sûre, l'attaquant est identifié, rendant ce type d'attaques peu exploitable.

6.2.3 Scénario d'usage d'adresse IP et de certificat d'attributs attribués par un serveur E-DHCP pour un contrôle d'accès

Dès que le client reçoit son adresse IP et son certificat d'attributs, il peut se connecter à l'intérieur ou en dehors du réseau donné.

Pour qu'un client puisse se connecter à un service donné à l'intérieur ou à l'extérieur du réseau donné, il doit dépasser le serveur de contrôle d'accès (voir figure 6-13). Les étapes à suivre sont les suivantes:

1. Le client se connecte, au serveur de contrôle d'accès, grâce à l'adresse IP qui lui a été affectée par le serveur *E-DHCP*.
2. Le client présente son certificat d'identité X.509 auprès du serveur de contrôle d'accès pour s'authentifier (on peut avoir une authentification SSL (*Secure Sockets Layer*) [RFC2246] des côtés client et serveur).
3. Le client présente son certificat d'attributs auprès du serveur de contrôle d'accès.
4. Le serveur de contrôle d'accès vérifie :
    a) Le certificat d'identité X.509 et plus précisément :
        - que la signature de l'autorité de certification est valide,
        - la date de validité du certificat d'identité,
        - la validité de la chaîne de certification,
        - …
    b) Le certificat d'attributs et plus précisément :

- que la signature de *l'AA* (*Attribute Authority)* est valide,
- la date de validité du certificat d'attributs,
- la validité du lien entre le certificat d'identité X.509 et son certificat d'attributs (voir figure 6-15). (Si le champ "*Holder*" du certificat d'attributs est égale à la valeur de *BaseCertificateID* du certificat d'identité X.509 du client).
- l'égalité entre:

  a) la valeur de l'adresse IP utilisée par le client pour se connecter au serveur de contrôle d'accès

  et

  b) la valeur de l'adresse IP contenue dans le certificat d'attributs.

Si la vérification dans la partie précédente est réussie, le serveur de contrôle d'accès permet au client d'accéder au service autorisé (référencé dans le champ extension de son certificat d'attributs).

Un scénario de contrôle d'accès est illustré dans la figure 6-13 :

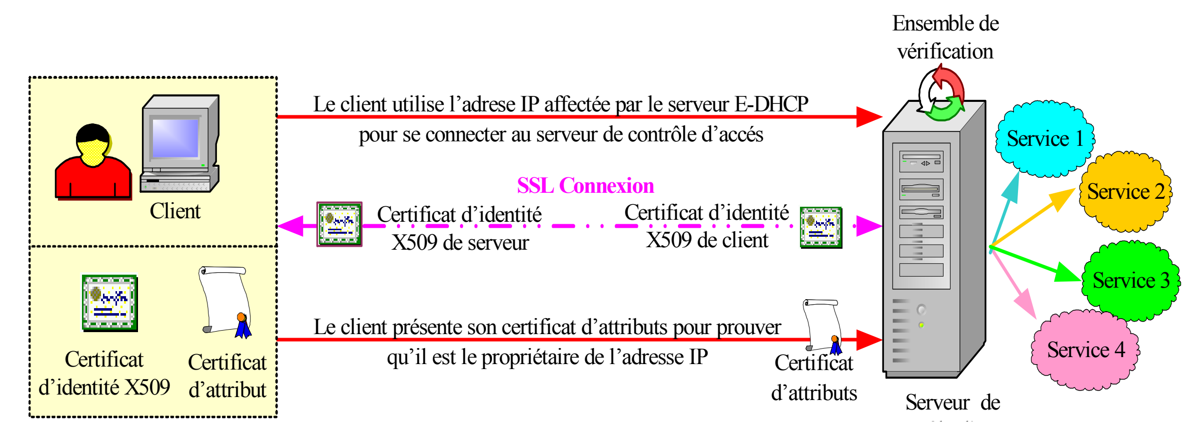


Figure 6-13 Scénario du contrôle d'accès

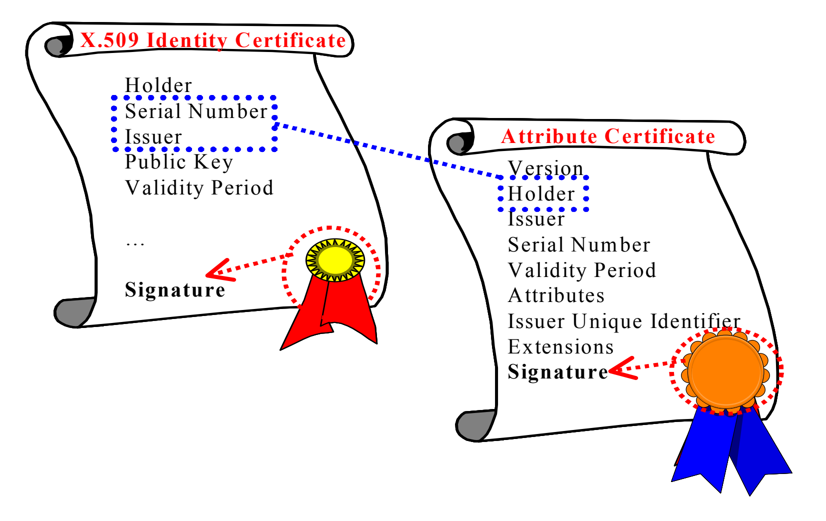


Figure 6-14 Lien entre certificat d'identité X.509 et certificat d'attributs

6.2.4 Les problèmes de sécurité résolus par la solution *E-DHCP*

On rappelle dans cette section les problèmes de sécurité classiques intrinsèques à *DHCP*. Nous allons voir dans quelle mesure *E-DHCP* [DEMERDraft] permet de régler ces problèmes. Nous nous intéressons d'abord aux attaques menées contre les serveurs *DHCP*, puis aux attaques menées contre les clients.

6.3.4.1 Famine des ressources que le serveur peut allouer :

Une attaque classique consiste à construire un client qui va demander un grand nombre d'adresses, jusqu'à ce que le stock du serveur *DHCP* soit épuisé. Ceci est possible si le client *DHCP* communique au serveur une adresse *MAC* différente à chaque requête. Il pourra malgré tout récupérer les messages de réponse du serveur *DHCP*, en utilisant par exemple le mode "*promiscuous*" [SAN] de sa carte réseau.

Cette attaque n'est pas possible avec *E-DHCP*, puisque chaque client doit s'authentifier au moyen de son certificat d'identité au début de la transaction.

6.3.4.2 Attaque par le milieu :

Une attaque de type "*man-in-the-middle*" consiste, pour un tiers pouvant intercepter les messages d'un client et d'un serveur *DHCP* classique, à falsifier les données transmises. Ainsi, il peut retransmettre au serveur *DHCP* les messages du client, en changeant uniquement l'adresse *MAC* source indiquée dans le message, et retransmettre au client les messages du serveur, en modifiant les données afin de lui transmettre de mauvais paramètres de connexion. Ceci empêchera le client de se connecter correctement au réseau, et permettra à l'attaquant de détourner autant qu'il le souhaite les communications du client, sans pour autant que celui-ci ne s'en rende compte. Lors d'une navigation web du client, par exemple, l'attaquant aura la possibilité de connaître toutes les informations échangées, y compris, notamment, les éventuels mots de passe. Ceci nous amène à dire qu'il est très important que la phase de connexion au réseau par *DHCP* soit inviolée.

Cette attaque ne peut plus fonctionner avec *E-DHCP* [ICETE] [SEC], puisque dès lors que l'attaquant falsifie les messages, il lui faut recréer une signature valide, chose dont il est incapable de réaliser, puisqu'il ne dispose pas de la clef privée de l'émetteur.

Une autre façon de mener une attaque par le milieu consiste à se faire passer pour le client et à envoyer au serveur un message *DHCPRELEASE* indiquant qu'on libère son adresse IP. Le client légitime ne sera bien entendu pas au courant et continuera d'utiliser cette adresse IP. Le serveur en revanche la réintègrera à son *pool* de ressources disponibles. *E-DHCP* évite ce type de problème, le faux client n'étant pas capable d'imiter la signature du client légitime. Il faut toutefois prévoir que le serveur n'autorise pas au client qui a utilisé *E-DHCP* par le passé à l'utiliser de nouveau (voire opter pour une configuration plus sévère : le serveur n'accepte que *E-DHCP*, et tout message *DHCP* standard pourrait être rejeté).

6.3.4.3 Faux serveur *DHCP* :

Une façon de généraliser l'attaque précédente de type "*man-in-the-middle*" consiste à créer un faux serveur *DHCP*. C'est une attaque particulièrement simple puisque tout individu connecté à un réseau est libre d'installer sur sa machine un serveur *DHCP* et de le configurer avec des valeurs erronées afin de tromper les clients potentiels.

Ceci n'est pas possible avec *E-DHCP*, car ce faux serveur devrait disposer d'un certificat reconnu valide par le client, et se servir de la clef privée associée. Cette situation illustre toutefois l'importance

que les certificats d'identité des serveurs autorisés sur le réseau soient, ou bien importés dans tous les clients avant leur première connexion, ou bien signés par une autorité de certification commune, afin que ceux-ci soient en mesure de vérifier la chaîne de confiance.

6.2.5 Intérêt de *E-DHCP*

Cette solution présente plusieurs avantages :

1. Elle évite la modification du protocole *DHCP* actuel.
2. Elle assure l'authentification du client, du serveur *DHCP* et du message *DHCP*, si l'authentification est souhaitée.
3. Elle utilise *RSA* comme mécanisme de signature digitale, ce qui assure une meilleure sécurité que celles des algorithmes de chiffrement symétrique. L'utilisation de ce mécanisme élimine les problèmes de flexibilité et de distribution de clefs existants au niveau de l'utilisation des clefs partagées [RFC3118].
4. Elle permet un contrôle strict des équipements en employant une authentification forte (en utilisant les certificats d'identité X.509 et les certificats d'attributs).
5. Elle n'est pas vulnérable à des attaques à travers l'interception des messages échangés (contrairement à la méthode d'authentification : *"Token Authentication"* [RFC3118]). En effet, même si un individu intercepte l'ensemble des quatres messages échangés, il ne pourra pas avoir une adresse IP du serveur *E-DHCP* puisque, quand on a implémenté *E-DHCP*, on a exigé des entités (client et serveur E-*DHCP*) de concatener à leur signature la valeur du champ "*xid*" afin de les chiffrer avec la clef publique du récepteur du message *DHCP* (voir la section 6.2.6.1).
6. Elle peut supporter l'authentification inter-domaines.
7. L'adresse IP est affectée à une identité et non pas à un terminal. Ainsi, *E-DHCP* identifie l'utilisateur plutôt que le terminal. L'identification du teminal est optionnelle.
8. Elle permet aux utilisateurs authentifiés d'obtenir une adresse IP de n'importe quel équipement relié au réseau. En d'autres termes, lorsqu'un utilisateur possède quelques équipements à relier au réseau, seulement l'utilisateur légitime sur chaque équipement peut exercer son droit d'obtenir et d'utiliser une adresse IP sur le système.
9. Le contrôle d'accès est basé sur le certificat d'attributs assurant la relation entre le client et son adresse IP qui lui est affectée. L'utilisation du certificat d'attributs confirme la possession du client de son adresse IP.
10. Les messages *DHCPDiscover* sont authentifiés par ce protocole (contrairement à [RFC3118]), ce qui rend le protocole invulnérable aux attaques de déni de services par le *flooding* par des messages *DHCPDiscover* non authentifiés.
11. Compatible avec *DHCPv6* (voir paragraphe 6.2.7).
12. Elle est une solution ouverte qui peut être généralisée pour être utilisée avec d'autres problèmes appropriés tels que *DHCP-NAT* (*Network Address Translation*).

6.2.6 Est-ce que *E-DHCP* est complètement sûr ?

La discussion qui suit vise à étudier la sécurité du protocole *E-DHCP* [ICETE] [SEC], dont on a vu les avantages par rapport au protocole de base *DHCP*, résolvant plusieurs attaques connues.

6.2.6.1 Résolution des attaques par le milieu

Les mécanismes de sécurité mis en place dans *E-DHCP* permettent de procéder à une authentification des parties en interdisant toute modification des messages par un attaquant. Toutefois, rien n'empêche l'attaquant de rejouer les messages qu'il a interceptés. Par exemple, si une station met fin à son bail et si l'attaquant a capturé les messages initiaux de négociation *DHCP*, il pourra procéder à la réallocation de la ressource libérée, en espérant que ce sera celle qui lui sera proposée par le serveur.

Afin d'éviter cela, nous avons proposé trois solutions :

- Demander aux entités (client/serveur) de dater leurs messages. Ceci empêche complètement le rejeu d'un message, puisque chaque message porte la date et l'heure de son envoi. Afin que ces données soient vérifiables, il faut que l'ensemble des serveurs et le parc des clients soient synchronisés sur la même horloge. Ceci est possible grâce à l'usage du protocole *NTP* [RFC1305]. D'autre part, il faut prendre en compte les temps réseaux qui font que, lors de sa réception, un message sera analysé à une heure de toute façon postérieure à son émission. Pour cette raison, l'émetteur peut inclure un second champ horaire, dans lequel il indique la durée de validité de son message à partir de la date d'émission. Cette marge permet de prendre en compte l'acheminement réseau. Une bonne valeur de ce champ serait de quelques secondes. Ce couple (*timestamp*, *marge*) est utilisé dans le protocole *TSIG* [RFC3645] qui permet à des acteurs *DNS* de signer les messages qu'ils échangent tout en empêchant le rejeu. Cette solution, élégante, présente toutefois un inconvénient majeur : elle nécessite une mise en œuvre du protocole *NTP*. Dans le cas d'une négociation *DHCP*, l'hypothèse que le client sera capable de réussir cette proposition est erronée : dans la plupart des cas, le client cherche à obtenir sa première adresse IP, et n'est donc pas encore capable de dialoguer avec un serveur de temps *NTP*...
- Demander aux entités (client/serveur) d'inclure un numéro de série dans leurs messages. Ce mécanisme permet de détecter le rejeu d'une série de messages, car en principe, entre deux messages, le compteur est incrémenté. Toutefois, dans le cas où plusieurs serveurs sont présents sur le site, chaque serveur ne peut pas effectuer un suivi des numéros de série de chaque client, puisqu'il n'est pas le seul interlocuteur de ceux-ci. De même, si un serveur perdait son fichier de suivi, une attaque de rejeu serait possible. Cette solution ne semble donc pas satisfaisante ;
- Utiliser le champ "*xid*" du paquet *DHCP*. [RFC2131] définit le champ "*xid*" comme étant l'identifiant de la transaction, initialisé à une valeur aléatoire par le client. Or, ce champ est optionnel, de plus, il peut être, selon les implémentations, incrémenté à chaque message ou bien à chaque session.

Dans le cas de *E-DHCP*, on a requéri la présence de champ "*xid*" et aussi son incrémentation à chaque échange pour éviter les attaques par rejeu. Certes, l'incrément à appliquer au champ "*xid*" est prévisible mais l'intégrité et l'authenticité du message sont garanties grâce à la signature (chiffrée ou pas) présente dans le champ "*Information d'Authentication*" du champ de l'option proposée.

On a exigé aux entités (client et serveur *E-DHCP*) de concatener à leur signature la valeur du champ "*xid*" afin de les chiffrer avec la clef publique du récepteur du message *E-DHCP*. Seulement le possesseur de la clef privée convenable (le récepteur) peut déchiffrer la valeur

contenue dans le champ "*Information d'Authentication*" et ainsi peut savoir la valeur du champ "*xid*".

6.2.6.2 Vérification des certificats

L'absence de connexion réseau permettant d'utiliser des ressources externes pose le problème de la récupération du certificat du serveur par le client non encore configuré. En effet, dans les messages échangés, ce n'est pas le certificat en lui-même qui est communiqué au client, mais l'*URI* permettant d'accéder à ce certificat. Il n'est pas trivial de le récupérer dans ce cas.

Si le client procède au renouvellement de ses ressources, ou bien s'il possède une seconde adresse IP, il peut suivre les *URI* pour récupérer les certificats nécessaires.

Si l'URI est le chemin (*path*) vers un fichier local sur la station cliente, alors le problème est réglé. Dans ce cas le client récupère le certificat d'identité du serveur en le demandant de l'administrateur du réseau.

Dans le cas où le client voudrait récupérer le certificat par le réseau, une solution pourrait être d'utiliser le protocole *TFTP* [RFC1350] de manière à demander une diffusion du certificat vers toutes les stations du sous-réseau. Cela est possible en positionnant l'adresse d'origine sur l'adresse IP 255.255.255.255, adresse vers laquelle le serveur renverra le certificat. Celui-ci sera ignoré par les autres stations étant donné qu'elles n'ont effectué aucune requête. Toutefois le risque de collision est important si deux machines effectuent simultanément un transfert. Et de toute façon, le transfert du fichier ne sera pas sécurisé, *TFTP* ne prévoyant pas de signer les messages de données échangés.

6.2.7 **Compatibilité de *E-DHCP* avec *DHCPv6***

IPv6 propose deux méthodes d'attributions d'adresses dynamiques (avec et sans état) qui ne sont pas exemptes de problèmes de sécurité. Des solutions existent pour parer à ces manquements mais celles-ci requièrent pour la plupart des infrastructures complémentaires (*AAA* [AAADraft], autorité de gestion de clef) ou des protocoles et options venant en complément (*IPSec*, *SEND* [SENDDraft], *CGA* [CGADraft], *ABK* [ABKDraft]).

DHCPv6 (*Dynamic Host Configuration Protocol for IPv6*) [RFC3315] est un service d'allocation d'adresses IPv6 (*Internet Protocol version 6*) [RFC2460] pour les terminaux (*hosts*). Il est à la fois le successeur de *DHCP* pour IPv4 et la forme avec-état de l'auto-configuration.

Tout le mécanisme d'autoconfiguration avec état est bâti sur le modèle du client-serveur et repose sur l'utilisation du protocole *DHCPv6*. La principale différence vient d'une simplification du code : le *DHCPv4* n'est qu'une extension du protocole *BOOTP* avec des fonctionnalités limitées. Les implémentations de *DHCPv4* sont conçues afin de gérer les adresses IP comme une ressource rare.

Dans le cadre de IPv6, les problèmes sont totalement différents et donc les protocoles devraient aussi l'être. [RFC2462] définit la configuration avec état comme un service qui attribue les adresses permanentes aux machines, donc plus comme *BOOTP* que *DHCP*. Cela conduit à un problème de fond : «La notion d'adresses temporaires (à la *DHCPv4*) a peu de sens en IPv6 ».

Il est néanmoins important de noter que *DHCPv6* a corrigé quelques problèmes de son prédécesseur (utilisation du *broadcast* massif) en s'appuyant sur des communications multi-destination (*multicast*) pour la découverte de serveurs d'un site. *DHCPv6* a ajouté quelques possibilités supplémentaires (méthode d'authentification "des entités et des messages *DHCP*", possibilité de reconfiguration des clients à partir du serveur, etc.).

Le mécanisme d'authentification de *DHCPv6* repose sur le même principe d'authentification de *DHCPv4* de IPv4 à savoir sur une option d'authentification (*Delayed Authentication Protocol)* [RFC3118]. Ce qui explique la raison pour laquelle *E-DHCP* [DEMERDraft] s'articule parfaitement avec *DHCPv6*. Cette extension de *DHCPv6* est possible par l'ouverture des options prévues de ce

protocole. *E-DHCP* garde la majorité de ces avantages par rapport à celle de *Delayed Authentication Protocol*.

6.3 Conclusion

Nous avons proposé dans ce chapitre une solution basée sur les certificats d'identité X.509 couplés avec les certificats d'attributs pour doter *DHCP* d'un mécanisme d'authentification forte tout en préservant sa dynamicité. Cette solution consiste à étendre le protocole *DHCP* en *E-DHCP* (*Extended Dynamic Host Configuration Protocol*). L'objectif est d'attribuer à l'équipement un certificat d'attributs contenant l'adresse Internet attribuée dynamiquement. Pour leur authentification au sein des architectures réseaux, les équipements peuvent faire preuve de l'authenticité de leur adresse en présentant leur certificat d'identification X.509 et le certificat d'attributs. Ainsi, le serveur *DHCP* est adossé à un serveur d'attribution de privilèges. Nous avons exposé le concept, le principe, les conditions d'usage, le mode de fonctionnement, le scénario d'usage et les avantages du *E-DHCP*. Ce protocole *DHCP* étendu possède des avantages notoires.

E-DHCP résout bien les problèmes de sécurité de *DHCP*, et permet d'effectuer un contrôle d'accès strict de l'utilisateur aux ressources du réseau grâce à l'utilisation des certificats d'attributs. Cette extension garantit l'authentification des entités (client et serveur *DHCP*) et l'intégrité des messages échangés. Elle utilise *RSA* comme support de signature électronique, ce qui est plus sûr que le *HMAC* proposé par [RFC3118].

Afin de faciliter l'implémentation de *E-DHCP*, nous présentons dans le chapitre suivant la modélisation *UML* du protocole *E-DHCP.*

Chapitre 7

7 Modélisation et implémentation de *E-DHCP*

Résumé

Après avoir présenté dans le chapitre précédent une extension du protocole *DHCP* à savoir *E-DHCP* (*Extended Dynamic Host Configuration Protocol*) et afin de valider notre proposition *E-DHCP* par une implantation à échelle réelle, nous présentons dans ce chapitre la modélisation *UML* et l'implémentation de *E-DHCP*. Dans la partie modélisation, nous présentons les *cas d'utilisation et* les *diagrammes de classes* du *DHCP* classique et de *E-DHCP.* Nous illustrons les diagrammes d'interactions entre un client *DHCP* et un serveur *DHCP* (respectivement entre un client *E-DHCP* et un serveur *E-DHCP*). Nous présentons également les diagrammes d'activités (du client et serveur *E-DHCP*) qui représentent les états de l'exécution d'une opération.

Dans la partie implémentation de *E-DHCP, nous* présentons l'implémentation de notre solution *E-DHCP* sur un client et sur un serveur *DHCP*. Pour ce faire, nous avons utilisé le logiciel *DHCP* proposé par l'*ISC* (*Internet Software Consortium*) sous licence *GPL* (*General Public License*). Nous présentons notre examen du code source de ce logiciel. Enfin, nous expliquons comment nous avons modifié les sources d'*ISC DHCP*, afin de rajouter l'option proposée par *E-DHCP*.

7.1 Modélisation *UML* de *E-DHCP*.

Nous présentons dans cette partie la modélisation *UML* du protocole *E-DHCP*.

7.1.1 Cas d'utilisation

Les cas d'utilisation servent à saisir le comportement attendu d'un système en cours de développement, sans avoir à préciser la façon dont ce comportement est réalisé.

Un cas d'utilisation décrit un ensemble de séquences d'actions qu'un système exécute pour produire un résultat tangible pour un acteur.

Un acteur représente un ensemble cohérent de rôles joués par les utilisateurs des cas d'utilisation en interaction avec ces cas d'utilisation [GuiUML].

La partie haute de la figure 7-1 représente les cas d'utilisation du *DHCP* classique.

L'acteur principal correspond au client *DHCP*; celui-ci dispose de quatre actions possibles avec le système:

- il peut "demander une configuration", cela correspond au message *DHCPDISCOVER*.
- il peut "sélectionner un serveur *DHCP*", par exemple lorsqu'un client *DHCP* reçoit des messages de *DHCPOFFER* de la part de serveurs *DHCP*, il peut sélectionner un serveur et une offre correspondante en lui répondant par *DHCPREQUEST*.
- il peut "récupérer une configuration", après réception d'un message *DHCPDACK*, le client *DHCP* peut se configurer.
- il peut "proposer une configuration", cela correspond au message *DHCPDINFORM*.

La partie basse du schéma représente les cas d'utilisation de *EDHCP*.

La solution *EDHCP* est une extension qui reste compatible avec le protocole *DHCP*. Ainsi, l'acteur principal de *EDHCP*, qui n'est autre que le client *EDHCP* hérite de l'acteur client *DHCP*; c'est-à-dire que toutes les actions possibles du client *DHCP* peuvent être exécutées par le client *EDHCP*.

Le client *EDHCP* a un pré-requis, c'est de disposer d'un certificat d'identité X509 (dont la chaîne de certifications est compatible avec le certificat d'identité serveur *EDHCP*).

Un certificat d'identité pouvant être associé à un équipement ou une personne, un client *EDHCP* a les même cas d'utilisation quelle que soit l'entité associée au certificat.

Ainsi dans notre représentation un "client *EDHCP* " peut être étendu indifféremment par un "équipement" (dans le cas où le certificat d'identité client est rattaché à une machine) ou une "personne" (dans le cas où le certificat d'identité identifie une personne).

Si le client *EDHCP* est spécialisé par une personne, celle-ci a la possibilité de "choisir un équipement".

Le client *EDHCP* héritant du client *DHCP* dispose donc des quatre actions définies pour le client *DHCP*.

Mais ces actions sont à leur tour étendues pour le client *EDHCP*.

- Lorsqu'un client *EDHCP* "demande une configuration", il communique l'*URI* de son certificat d'identité.
- Lorsqu'un client *EDHCP* "sélectionne une offre d'un serveur *EDHCP* " il "récupère le certificat d'identité du serveur *EDHCP*".
- L'action "Récupérer une configuration" est étendue par "récupérer un certificat d'attributs".
- De même "proposer une configuration" est étendue par "envoyer un certificat d'attributs".

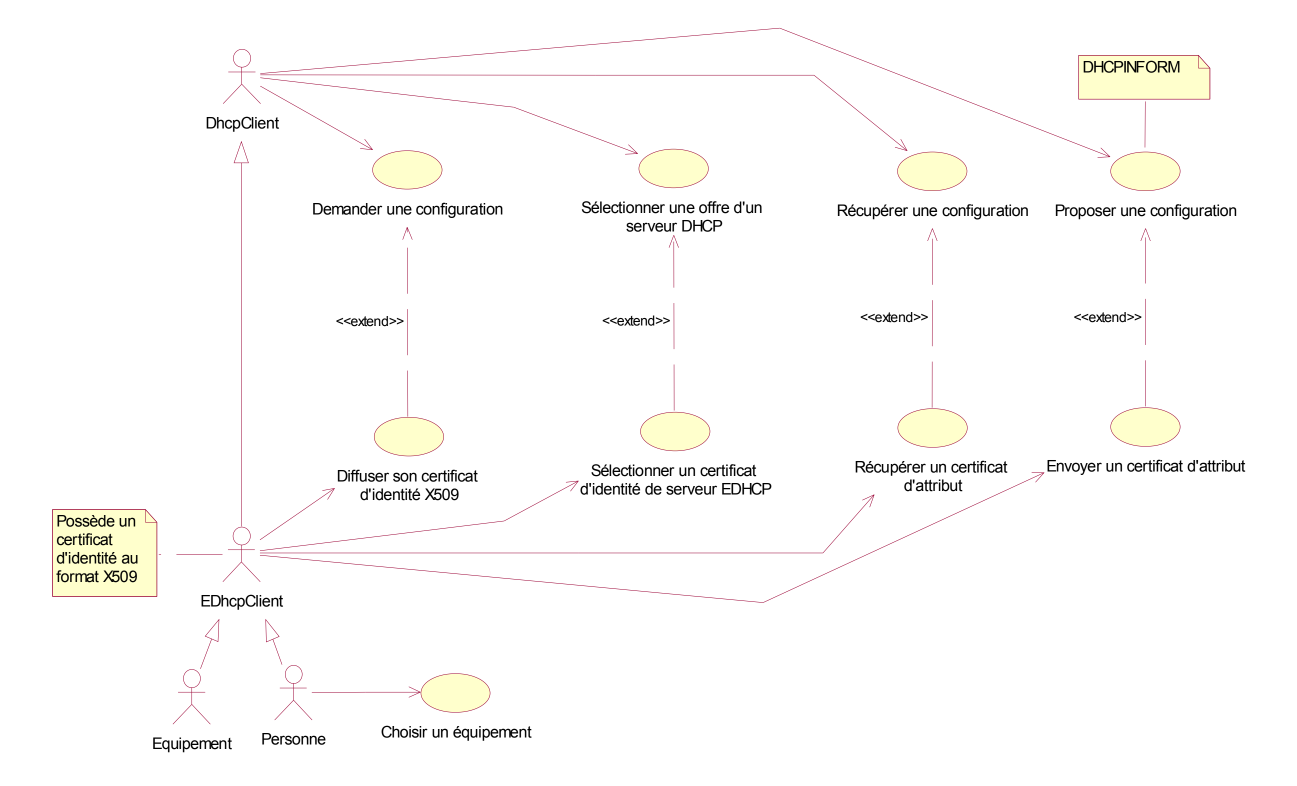


Figure 7-1 Diagramme des cas d'utilisation de *E-DHCP*

7.1.2 Diagramme de classes

Les diagrammes de classes sont les diagrammes les plus courants dans la modélisation des systèmes orientés objets. Ils représentent un ensemble de classes, d'interfaces et de collaborations, ainsi que leurs relations de dépendance, de généralisation et d'association [GuiUML].

7.1.2.1 Extension de DHCP par EDHCP

Avec un premier diagramme (voir figure 7-2), nous représentons les classes d'un système *DHCP* classique et de l'extension *EDHCP* ainsi que leurs relations d'héritage.

La partie haute du diagramme (voir figure 7-2) présente les classes d'un système client/serveur *DHCP* classique. Ainsi les classes *ClientDhcp* et *ServeurDhcp* disposent de méthodes pour construire les différents messages Dhcp (*construitDhcpDiscover*(), *construitDhcpOffer*(), ...). Ces deux classes sont liées par une association "communique" bidirectionnelle.

C'est une association de classe; un *ClientDhcp* et un *ServeurDhcp* "communiquent" par l'intermédiaire d'un "*PaquetDhcp*". De plus un *ClientDhcp* peut "communiquer" avec plusieurs *ServeurDhcp*, et vice versa.

Pour "communiquer", un *ClientDhcp* et un *ServeurDhcp* doivent implémenter des méthodes communes définies par l'interface "*ManipuleDhcp*" (comme *envoiePacketDhcp()*, *récupèrePacketDhcp()* ...).

Le but de la solution *EDHCP* étant d'étendre un système *DHCP*, nous définissons les classes *ClientEDhcp* et *ServeurEDhcp* par héritage des classes *ClientEDhcp* et respectivement *ServeurEDhcp*. L'extension *EDHCP* est représentée dans la partie basse du diagramme.

De même les *ClientEDhcp* et *ServeurEDhcp* sont liés par une association "communique" déduite implicitement de la relation entre les classes parentes *ClientDhcp* et *ServeurDhcp*.

Un paquet *EDHCP* est un paquet *DHCP* avec des options spécifiques, ainsi on définit la classe d'association PaquetEDhcp par héritage de PaquetDhcp, et il est lié à l'association "communique" des *ClientEDhcp* et *ServeurEDhcp*.

Pour manipuler le champ *option* d'un paquet *EDHCP*, les classes *ClientEDhcp* et *ServeurEDhcp* nécessitent des méthodes communes (*signeChiffreEDhcp()*, *vérifieCertificat()*, ...) qui sont spécifiées par l'interface *ManipuleEDhcp*.

L'interface *ManipuleEDhcp* implémentée par *ClientEDhcp* et *ServeurEDhcp* étend l'interface *ManipuleDhcp* implémentée par *ClientEDhcp* et *ServeurEDhcp*.

7.1.2.2 Certificats et cryptographie pour EDHCP

Ce diagramme (voir figure 7-3) est une vue des classes des outils de cryptographie et de gestion des certificats nécessaires au système *EDHCP*.

En effet pour pouvoir construire ou lire l'option d'authentification d'un paquet *EDHCP*, les clients et serveurs *EDHCP* doivent disposer de fonctions de cryptographie standardisées. Nous avons donc défini une interface *LibrairieCrypto* qui regroupe des fonctions de chiffrement, déchiffrement, signature, vérification de signature et certificat, ainsi que de récupération de certificats.

L'implémentation de l'interface *LibrairieCrypto* peut être basée aussi bien sur une librairie standard de fonctions cryptographiques (de type openssl par exemple) que sur des commandes systèmes, peu importe. Il y a donc une relation d'"utilisation" entre les classes client/serveur *EDHCP* et l'interface *LibrairieCrypto*.

La classe abstraite *BDCertificat* représente une base de données de certificat générique, elle est une association de composition (agrégation forte) avec la classe abstraite Certificat qui représente un certificat générique.

La multiplicité est définie de telle sorte qu'un Certificat ne doit appartenir qu'à une seule *BDCertificat*, et qu'une *BDCertificat* peut ne contenir aucun certificat, ou en contenir une multitude de Certificat. *BDCertificat* et Certificat sont identifiés par une *URI* définie en attribut publique.

Une *BDCertificat* peut être spécialisée en *BDCertIdentité* ou *BDCertAttribut*. De même, un Certificat peut être spécialisé en *CertificatX509* ou *CertificatAttribut*.

Ainsi, une *BDCertIdentité* est composée de CertificatX509 tandis qu'une *BDCertAttribut* est composée de *CertificatAttribut*.

L'association "litCertificat" modélise le fait que l'interface *ManipuleCrypto* accède en lecture aux certificats d'une *BDCertificat* au moyen du qualificateur certificat de type *URI*.

L'interface *ManipuleAA* a de plus une association similaire, mais utilisée pour l'accès en écriture à une *BDCertAttributs*

L'interface *ManipuleAA* peut être "utilisée" par la classe *ServeurEDhcp*, ainsi seuls les serveurs *EDHCP* peuvent construire ou révoquer des certificats d'attributs.

Cette interface peut être implémentée aussi bien par une librairie *AA* que par un serveur *AA* dans le cas où le serveur *EDHCP* fait appel à un serveur *AA* externe au système.

7.1.2.3 Vue synthétique de EDHCP

Cette vue (voir figure 7-4) permet de résumer le fonctionnement d'un système *EDHCP*. Clients et serveurs *EDHCP* peuvent lire des certificats d'identité X509 ou des certificats d'attributs qualifiés par une *URI*. De plus, un serveur *EDHCP* a la capacité de construire des certificats d'attributs.

Ces associations sont déduites implicitement par l'"utilisation" des interfaces *ManipuleCrypto* & *ManipuleAA*, par l'"héritage" des classes *BDCertIdentité* & *BDCertAttribut* sur *BDCertificat*, et par la "composition" de celles-ci respectivement par les classes *CertificatX509* & *CertificatAttribut*. Ainsi, un client et serveur *EDHCP* "communiquent" par des paquets *EDHCP*.

7.1.2.4 Vue globale des méthodes EDHCP

Ce schéma (voir figure 7-5) est une vue qui synthétise les associations d'héritage, d'implémentation et d'utilisation des classes *ClientEDhcp* et *ServeurEDhcp*, mettant ainsi en valeur toutes les méthodes directement associées.

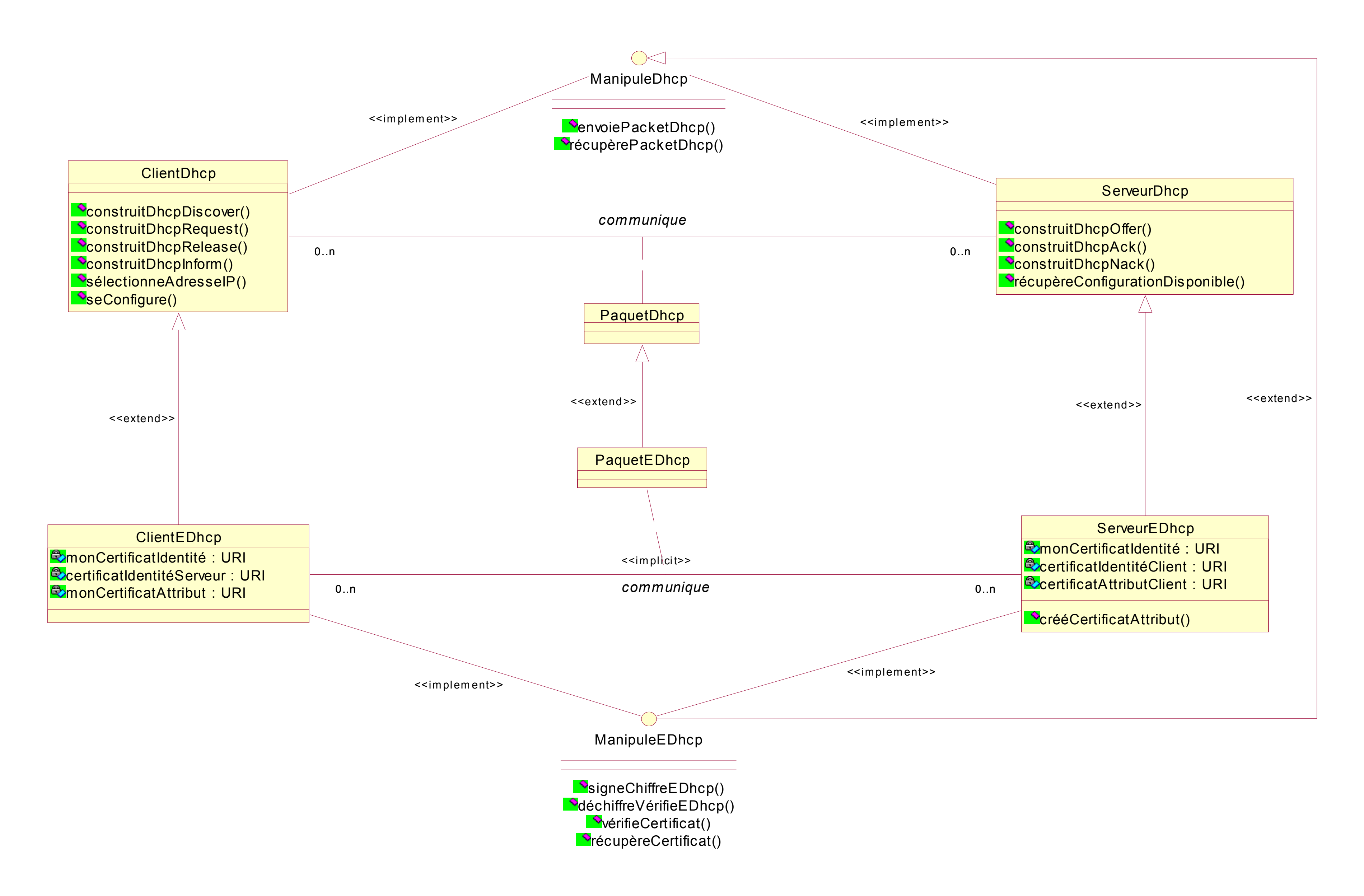


Figure 7-2 Daigramme de classes – Extension de *DHCP* par *E-DHCP*

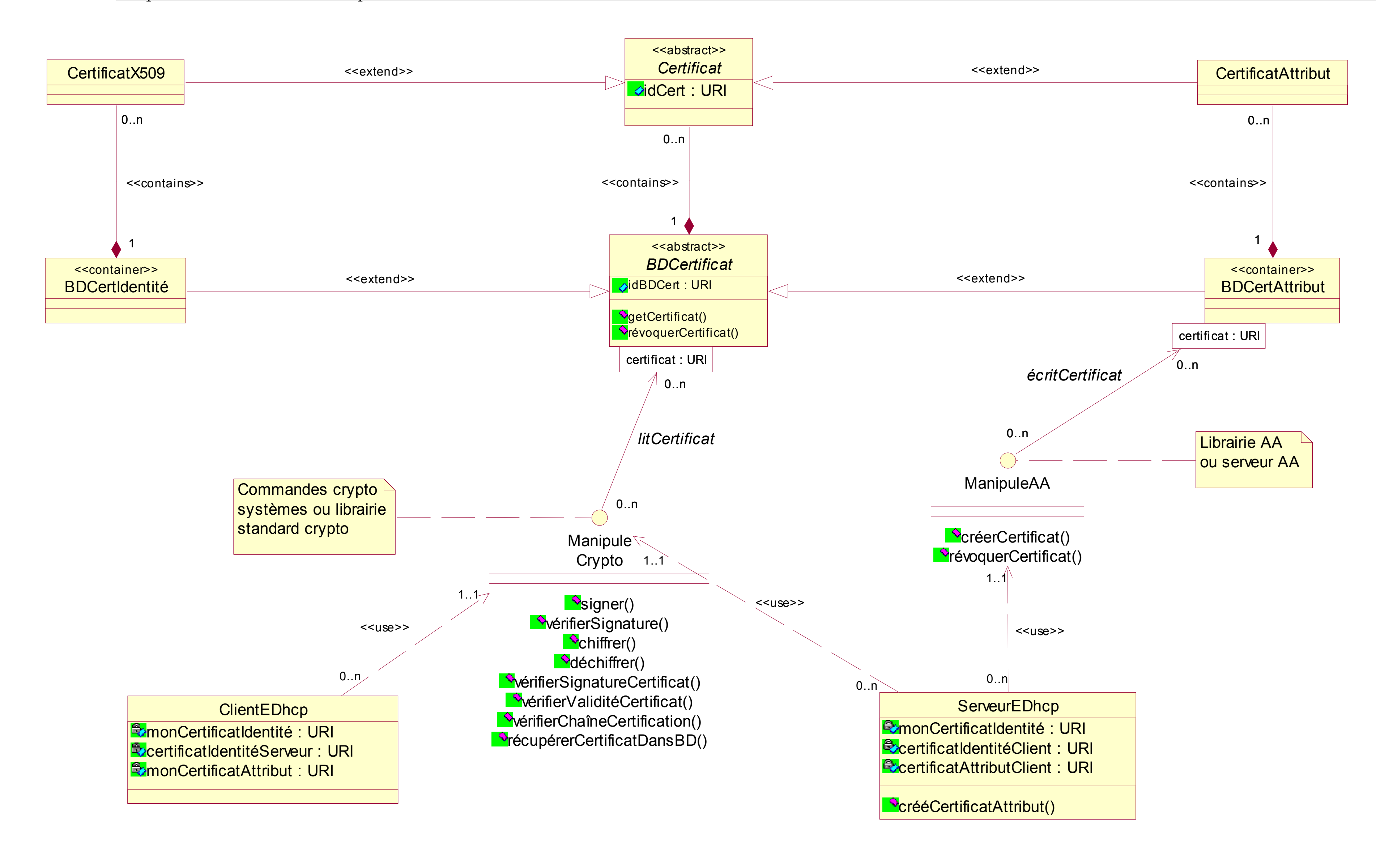


Figure 7-3 Diagramme de classes – Certificat et cryptographie pour *E-DHCP*

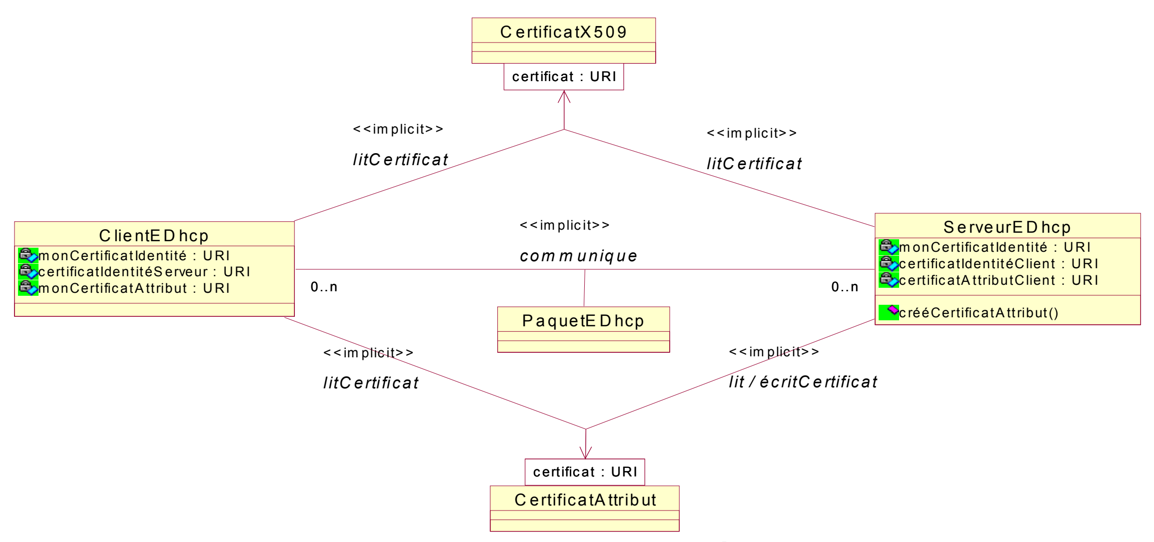


Figure 7-4 Diagramme de classes – Vue synthétique d'*E-DHCP*

ClientDhcp
constuitDhcpDiscover()
construitDhcpRequest()
construitDhcpRelease()
construitDhcpInform()
sélectionneAdresseIP()
seConfigure()

<<implement>>
ManipuleDhcp
envoiePacketDhcp()
récupèrePacketDhcp()
<<implement>>

ServeurDhcp
construitDhcpOffer()
construitDhcpAck()
construitDhcpNack()
récupèreConfigurationDisponible()

<<extend>>
<<extend>>

ClientEDhcp
monCertificatIdentité : URI
certificatIdentitéServeur : URI
monCertificatAttribut : URI

<<implement>>
ManipuleEDhcp
signeChiffreEDhcp()
déchiffreVérifieEDhcp()
vérifieCertificat()
récupèreCertificat()
<<implement>>

ServeurEDhcp
monCertificatIdentité : URI
certificatIdentitéClient : URI
certificatAttributClient : URI
crééCertificatAttribut()

0..n
<<use>>
1..1
0..n
<<use>>
1..1
0..n
<<use>>
1..1

ManipuleCrypto
signer()
vérifierSignature()
chiffrer()
déchiffrer()
vérifierSignatureCertificat()
vérifierValiditéCertificat()
vérifierChaîneCertification()
récupérerCertificatDansBD()

<<implement>>
<<library>>
LibrairieStandardCrypto

<<implement>>
CommandesSytèmeCrypto

ServeurAA
<<implement>>
ManipuleAA
créerCertificat()
révoquerCertificat()

<<implement>>
<<utility>>
LibrairieStandardAA

Figure 7-5 Diagramme de classses – Vue globale de méthode *E-DHCP*

7.1.3 Diagrammes d’interactions

Une interaction est un comportement qui comprend un ensemble de messages échangés au sein d'un groupe d'objets, dans un contexte particulier, pour atteindre un objectif.

Un message est la spécification d'une communication entre objets, qui transporte des informations et qui s'effectue dans le but de déclencher une activité [GuiUML].

Un diagramme d'interactions montre une interaction, c'est-à-dire un ensemble d'objets et leurs relations, ainsi que les messages qui peuvent circuler entre eux.

Il existe deux types de diagramme d'interaction :

1. *le diagramme de séquence* (voir figure 7-6, 7-7, 7-8, 7-11 et 7-12) qui met l'accent sur le classement des messages par ordre chronologique, c'est un tableau dans lequel les objets sont rangés le long de l'axe des abscisses et les messages, par ordre croissant d'apparition, le long de l'axe des ordonnées.
2. *le diagramme de collaborations* (voir figure 7-9, 7-10 et 7-13) qui met en évidence l'organisation structurelle des objets qui envoient et reçoivent des messages, c'est un ensemble de sommets et d'arcs.

7.1.3.1 Interaction entre client et serveur **DHCP** *(respectivement* **EDHCP)**

Dans cette partie, nous représentons les interactions entre un client *DHCP* et un serveur *DHCP* (respectivement entre un client *EDHCP* et un serveur *EDHCP*) pour une *diagramme de séquence* de configuration du client par le serveur.

1.Configuration *DHCP* (voir figure 7-6) : Pour faire face au flot de messages *DHCP*, concentrons nous d'abord sur le cas d'une configuration *DHCP* classique. Le client commence par appeler une méthode pour construire un paquet *DHCP* de type *DHCPDISCOVER*. Par une autre méthode, il envoie ce paquet au serveur *DHCP*. Après avoir récupéré ce paquet, le serveur choisit une configuration disponible pour le client et lui répond par un *DHCPOFFER*. Ensuite, le client sélectionne une adresse de l'offre et envoie sa sélection par un *DHCPREQUEST*. Le serveur accepte par un *DHCPACK*. Enfin le client se configure.

2.Configuration *EDHCP* (voir figure 7-7, 7-8, 7-9 et 7-10) : Dans cette séquence on retrouve tous les messages de la séquence *EDHCP*. La principale différence est qu’avant d'envoyer un paquet *DHCP*, l'émetteur signe le paquet et chiffre cette signature; et respectivement à la réception d'un paquet *DHCP*, le récepteur déchiffre la signature et la vérifie. De plus, dans cet exemple nous avons fait l'hypothèse que client et serveur ne se connaissent pas, donc lors du premier échange ils doivent récupérer et vérifier le certificat d'identité de leur correspondant.

Les méthodes utilisées pour chiffrer, signer, récupérer et vérifier un certificat seront décrites dans les sections suivantes.



Figure 7-6 Diagramme de séquence – configuration *DHCP*

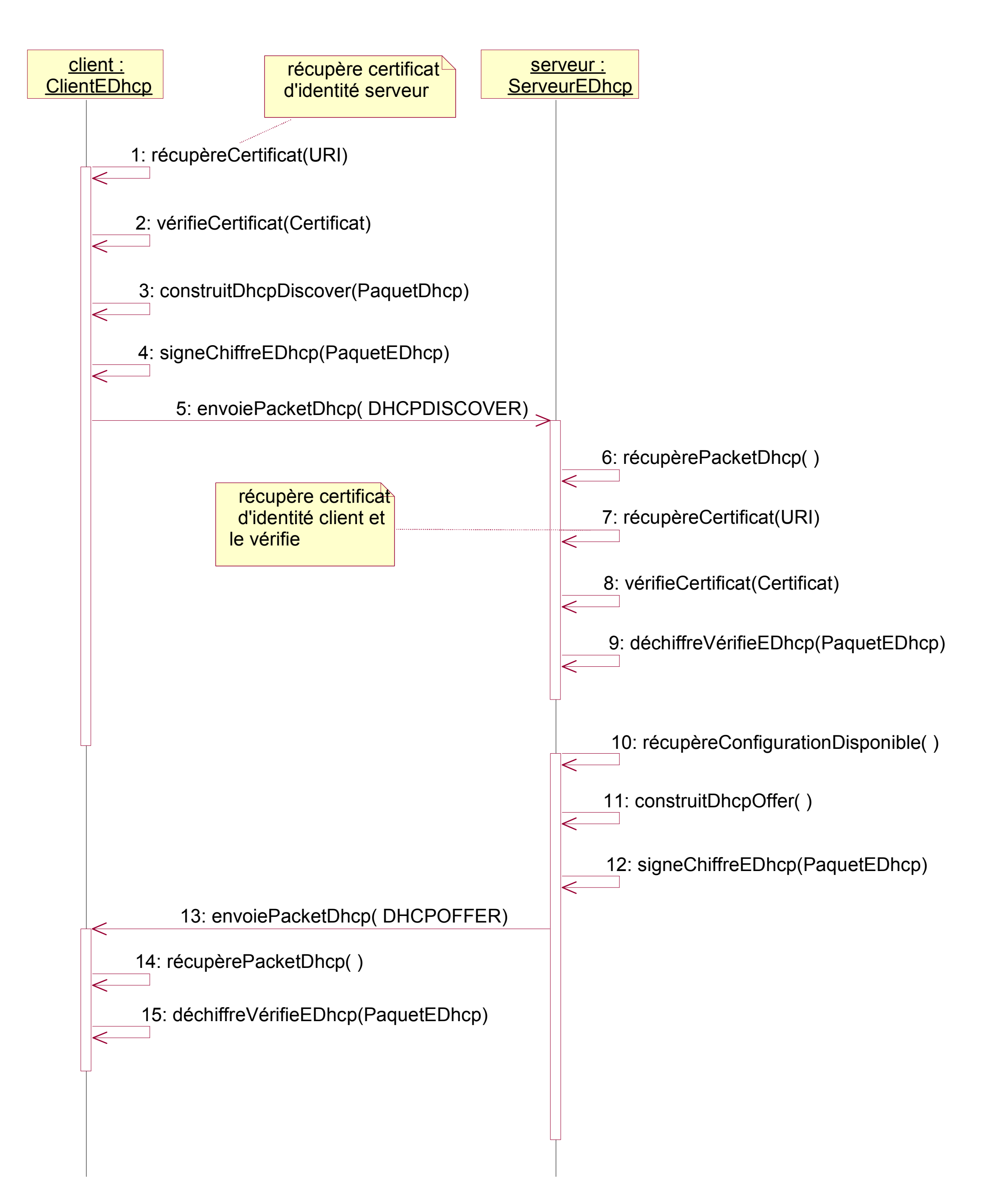


Figure 7-7 Diagrammme de séquence – configuration *E-DHCP* – partie 1/2

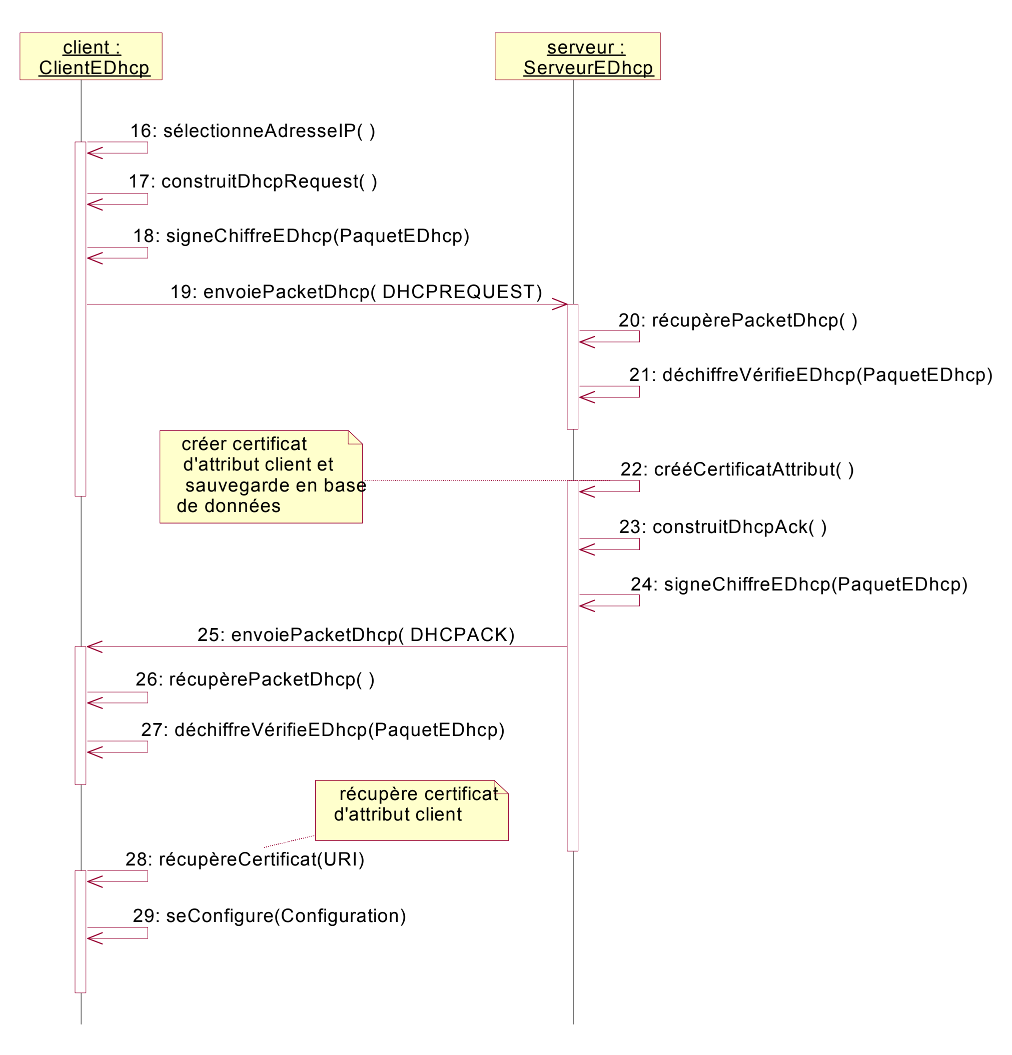


Figure 7-8 Diagramme de séquence – configuration *E-DHCP* – partie 2/2

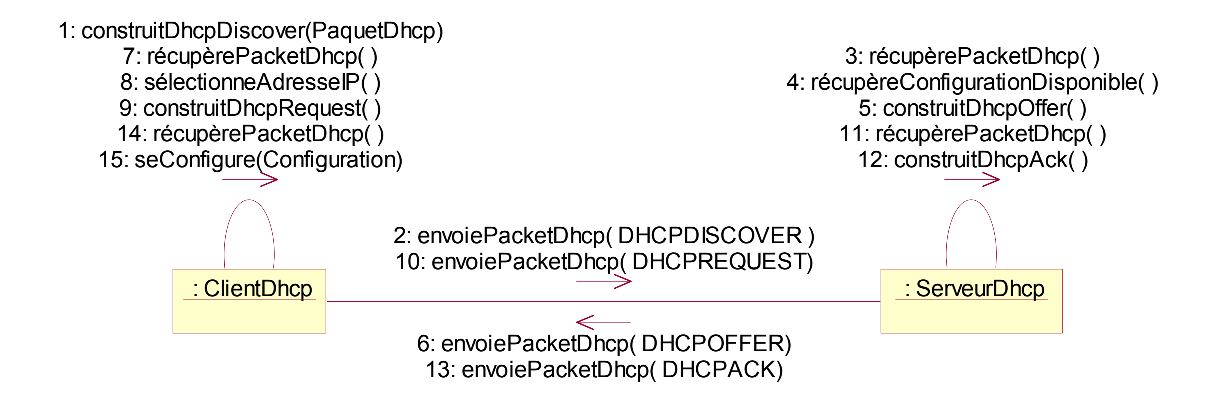


Figure 7-9 Diagramme des collaborations – configuration *DHCP*

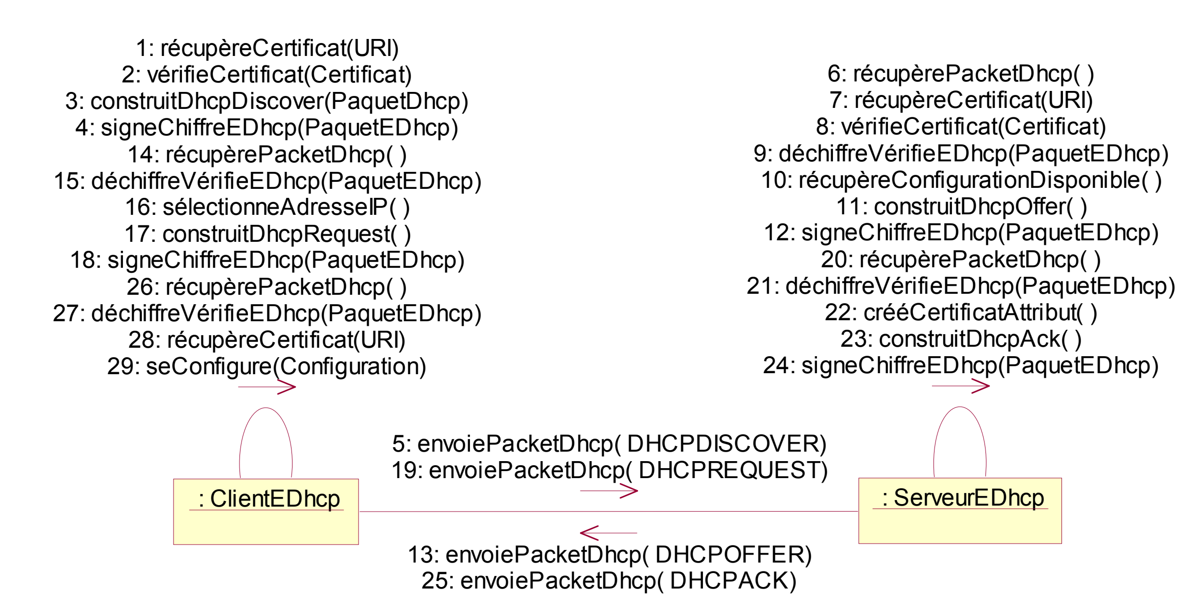


Figure 7-10 Diagramme des collaborations – configuration *E-DHCP*

Les diagrammes de collaborations pour *DHCP* et pour *EDHCP*, générés automatiquement à partir des diagrammes de séquences, mettent en valeur les envois des paquets *DHCPDISCOVER*, *DHCPOFFER*, *DHCPREQUEST* et *DHCPACK* entre client et serveur *DHCP* ou *EDHCP*.

7.1.3.2 Récupération de certificat et crypto paquet EDHCP

Dans cette partie nous présentons deux diagrammes de séquence d'un client ou d'un serveur *E-DHCP*. Le premier diagramme présente les séquences à suivre par un client ou un serveur *E-DHCP* pour récupérer à partie d'un *URI* un certificat et pour vérifier la validité du certificat. Le deuxième diagramme présente les séquences à suivre par un client ou un serveur *E-DHCP* pour signer et chiffrer un paquet *DHCP*.

1. Récupération et vérification d'un certificat (voir figure 7-11 et 7-13) : Pour récupérer un certificat à partir de son URI, un client fait appel à sa librairie de cryptographie qui interroge une base de données de certificats et retransmet le certificat au client.

   Pour vérifier un certificat, un client demande séquentiellement à sa librairie de cryptographie de vérifier la signature du certificat, de vérifier sa date de validité et enfin, de vérifier sa chaîne de certification (opération qui demande plus de temps).

   La même séquence est valable pour un serveur.

2. Signature et chiffrement d'un paquet *EDHCP* (voir figure 7-12 et 7-14) : Après avoir construit le paquet *DHCP* à envoyer, l'expéditeur appelle la fonction récursive *signeChiffreEDhcp*(). Celle-ci fait appel aux méthodes de signature et de chiffrement de sa librairie de cryptographie.

   De même après avoir récupéré un paquet *DHCP*, le récepteur appelle la fonction récursive *déchiffreVérifieEDhcp*(). Celle-ci fait appel aux méthodes de déchiffrement et de vérification de signature de sa librairie de cryptographie.

   Remarquons que par commodité dans cet exemple, client et serveur font appel à la même librairie.

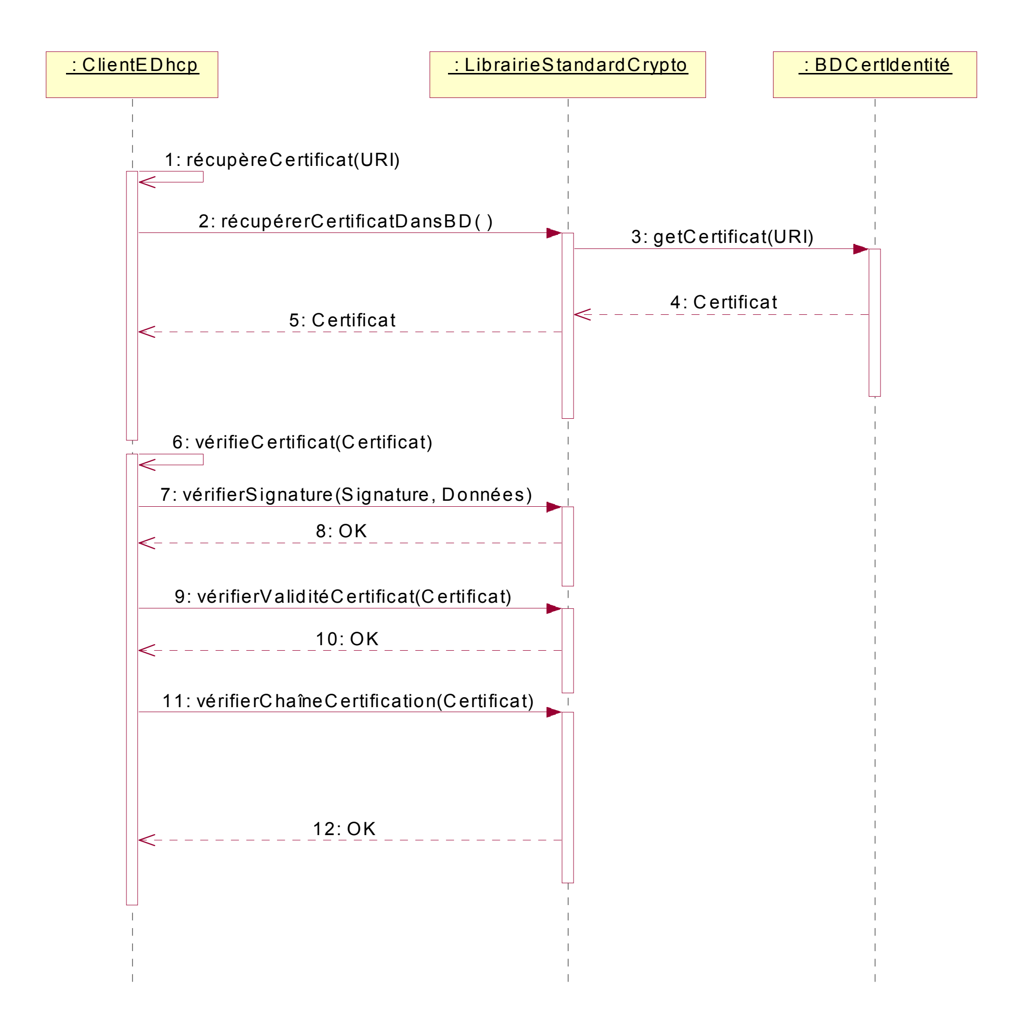


Figure 7-11 Diagramme de séquence – récupération et vérification d'un certificat

: ClientEDhcp
: LibrairieStandardCrypto
: ServeurEDhcp
1: construitDhcpDiscover(PaquetDhcp)
2: signeChiffreEDhcp(PaquetEDhcp)
3: signer(Données)
4: Signature
5: chiffrer(Données)
6: DonnéesChiffrées
7: envoiePacketDhcp(PaquetDhcp)
8: récupèrePacketDhcp( )
9: déchiffreVérifieEDhcp(PaquetEDhcp)
10: déchiffrer(DonnéesChiffrées)
11: Données
12: vérifierSignature(Signature, Données)
13: OK

Figure 7-12 Diagrammme de séquence – signature et chiffrement d'un paquet *E-DHCP*

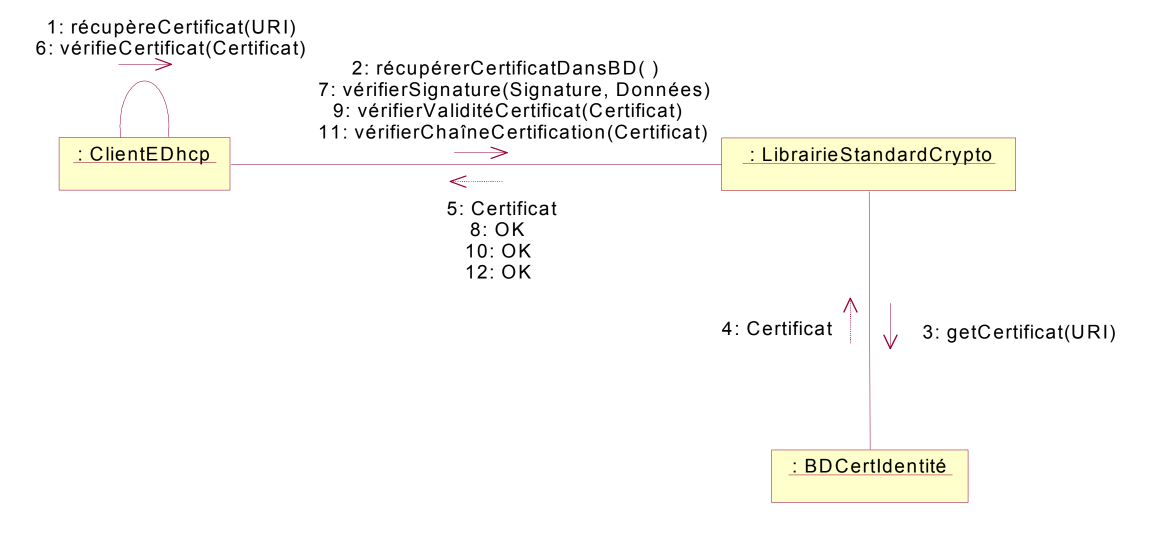


Figure 7-13 Diagramme des collaborations – récupération et vérification d'un certificat

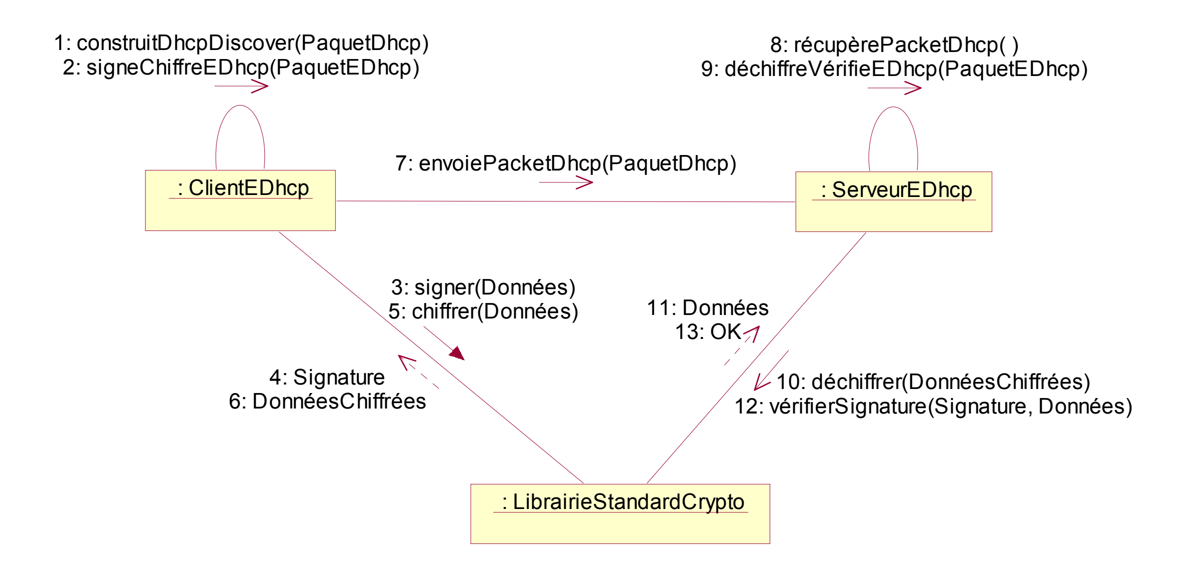


Figure 7-14 Diagramme des collaborations – siganture et chiffrement d'un paquet *E-DHCP*

7.1.4 Diagrammes d'activités

Un diagramme d'activités représente les états de l'exécution d'une opération ; chaque opération peut être définie par des sous-opérations représentées par des états d'activité du diagramme [DUNL]. Une activité est une exécution non atomique en cours à l'intérieur d'un automate à états finis [GuiUML]. Tandis qu'un diagramme d'états est particulièrement utile pour décrire la dynamique d'un système déterministe (C'est à dire qu'à tout moment, il ne peut y avoir, au plus, qu'un seul changement d'états possible ; le tout est de déterminer toutes les transitions possibles) [DUNL].

7.1.4.1 Serveur

- **Diagramme d'activités du serveur *E-DHCP***

Après le démarrage d'un serveur *E-DHCP*, celui-ci se met en attente de messages. A l'arrivée d'un message, une tâche est lancée, en parallèle, pour traiter le message (voir figure 7-15).

Cette tâche s'occupe tout d'abord de désencapsuler l'option *E-DHCP*, puis, en cas de succès, de traiter le message. La désencapsulation de l'option *E-DHCP* consiste à extraire la valeur de l'option, puis, selon la valeur du champ "*Flag*", à déchiffrer (si *Flag* vaut 1) et à vérifier la signature présente dans le champ "*Authentication Information*".

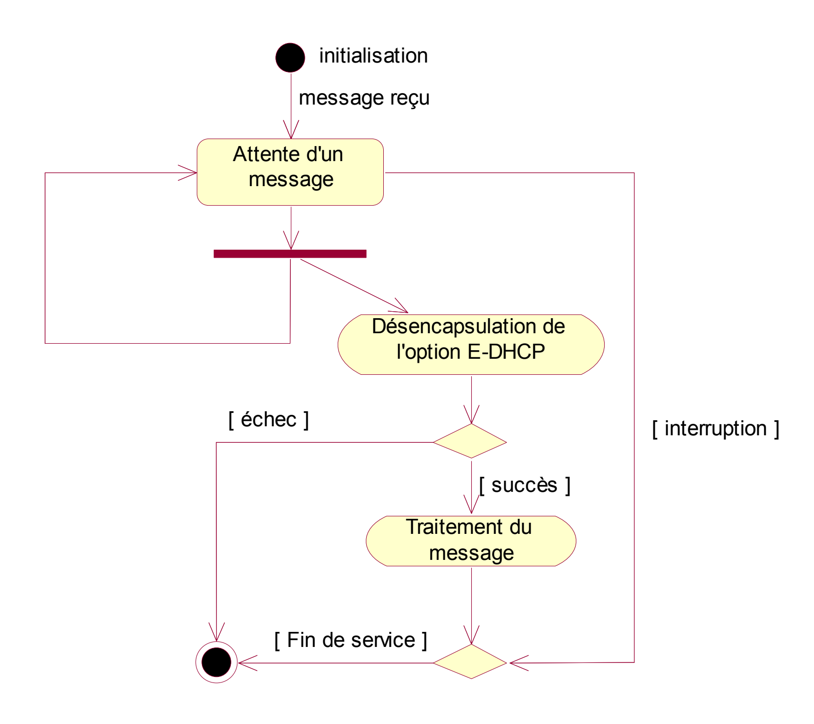


Figure 7-15 Diagramme d'activités du serveur *E-DHCP*

➢ **Diagrammes d'états : « Désencapsulation de l'option *E-DHCP* »**

Après qu'une entité (serveur ou client) désencapsule l'option *E-DHCP* depuis un message *DHCP*, elle récupère le certificat X.509 de l'émetteur du message, grâce au champ "*URIIdentityCertificate*" (voir figure 7-16).

Dans le cas où le champ "*Flag*" vaut 1, l'entité déchiffre le champ "*Authentication Information*" avec sa clef privée.

A ce stade, le champ "*Authentication Information*" contient la signature de tout le message DHCP reçu, avec la clef privée de l'émetteur du message. L'entité va vérifier l'intégrité du message reçu, en validant sa signature. Le détail concernant la validation de la signature figure dans le diagramme ci-dessous.

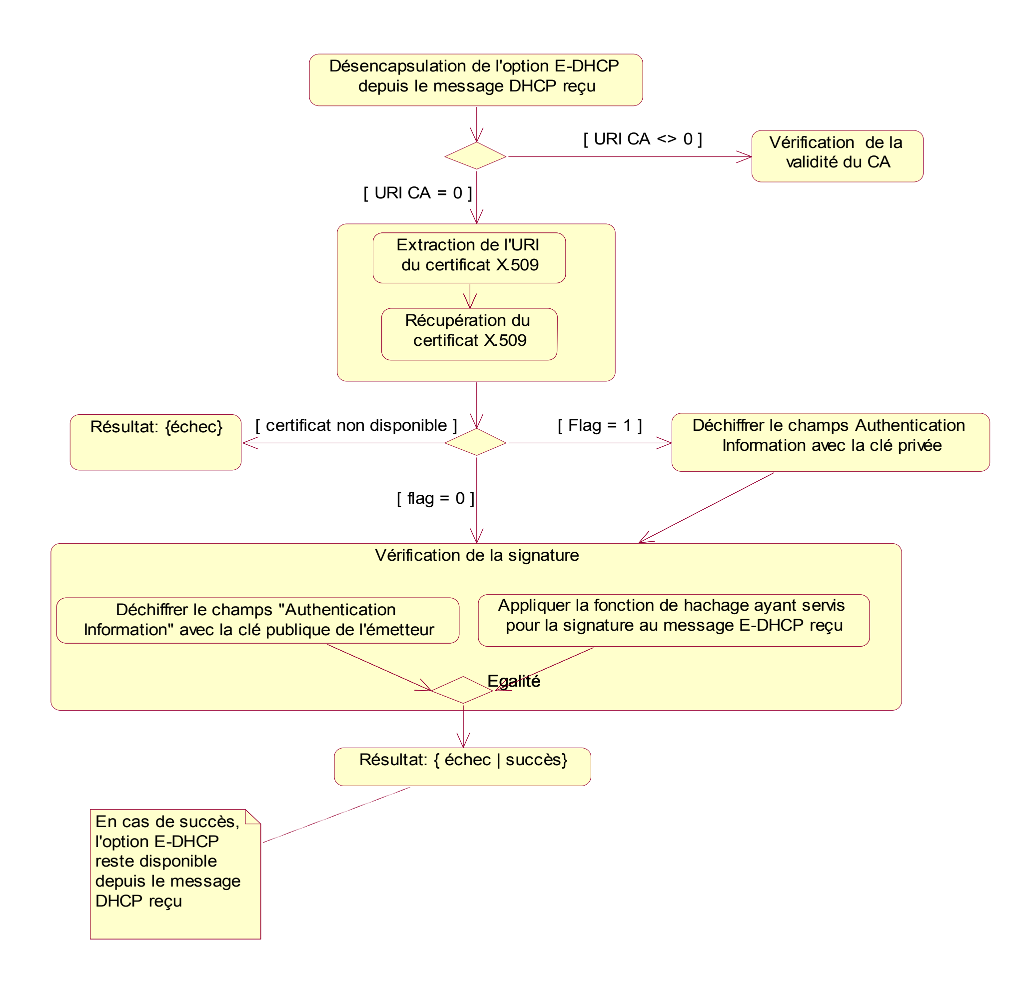


Figure 7-16 Diagramme d'états et transitions : Désencapsulation de l'option *E-DHCP*

➢ Diagramme d'états : « Traitement du message *DHCPDISCOVER* »

Après l'authentification du message *DHCPDISCOVER* , si le serveur dispose d'une adresse IP pour le client, il encapsule l'option d'authentification *E-DHCP* dans un message *DHCPOFFER* puis envoie le message et mémorise l'offre faite au client (voir figure 7-17).

Il faut noter, qu'à ce niveau, la signature du message s'applique à tout le message *DHCPOFFER*, sauf les champs "*hops*" et "*giaddr*".

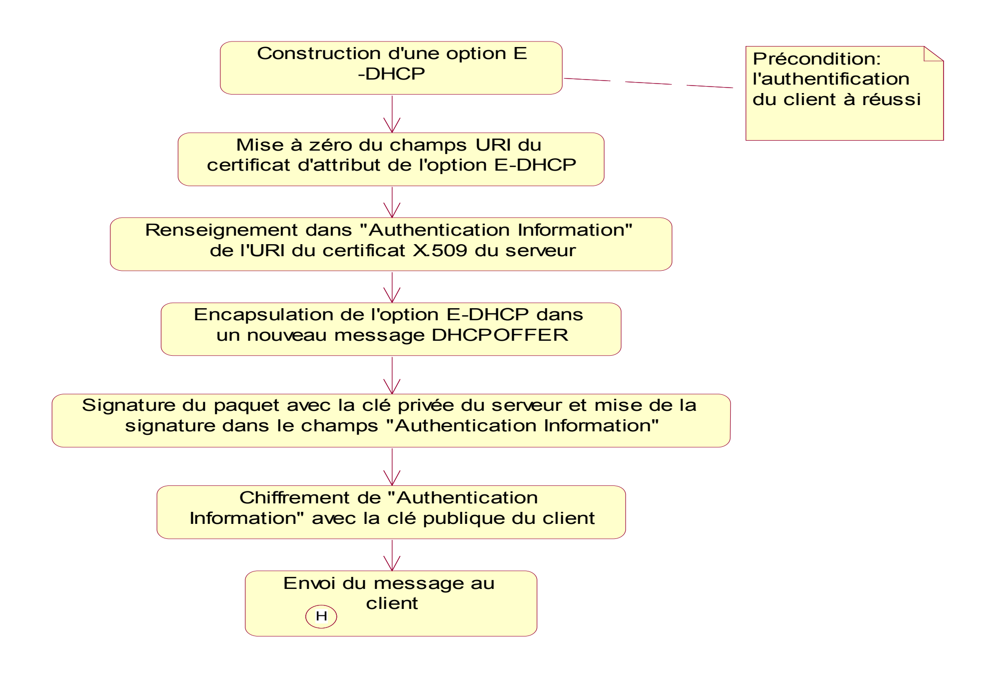


Figure 7-17 Diagramme d'états et transitions : Traitement du message *DHCPDISCOVER*

➢ Diagramme d'états : « Traitement du message *DHCPREQUEST* »

A la réception d'un message *DHCPREQUEST*, et après désencapsulation de l'option *E-DHCP*, deux cas de figures se présentent (voir figure 7-18):

– L'option *E-DHCP* a un champ "*URIAttributeCertificate*" nul : le serveur tente de restaurer l'offre faite précédemment au client ; en cas d'échec, le serveur transmet un message *DHCPNACK* avec authentification *E-DHCP*.

– L'option *E-DHCP* a un champ "*URIAttributeCertificate*" non nul : le serveur vérifie que le certificat d'attributs référencé par ce champ est valide, dans ce cas, le client souhaite renouveler son bail ; le serveur *E-DHCP* (ayant les attributions d'une autorité d'attributs) crée un nouveau certificat d'attribut puis le stocke dans la base des certificats d'attributs. Sinon, le serveur transmet un message *DHCPNACK* avec authentification *E-DHCP*.

Après la création du nouveau certificat d'attributs, le serveur encapsule l'option d'authentification *E-DHCP* (avec l'URI du nouveau certificat d'attributs) dans un message *DHCPACK* et le transmet au client.

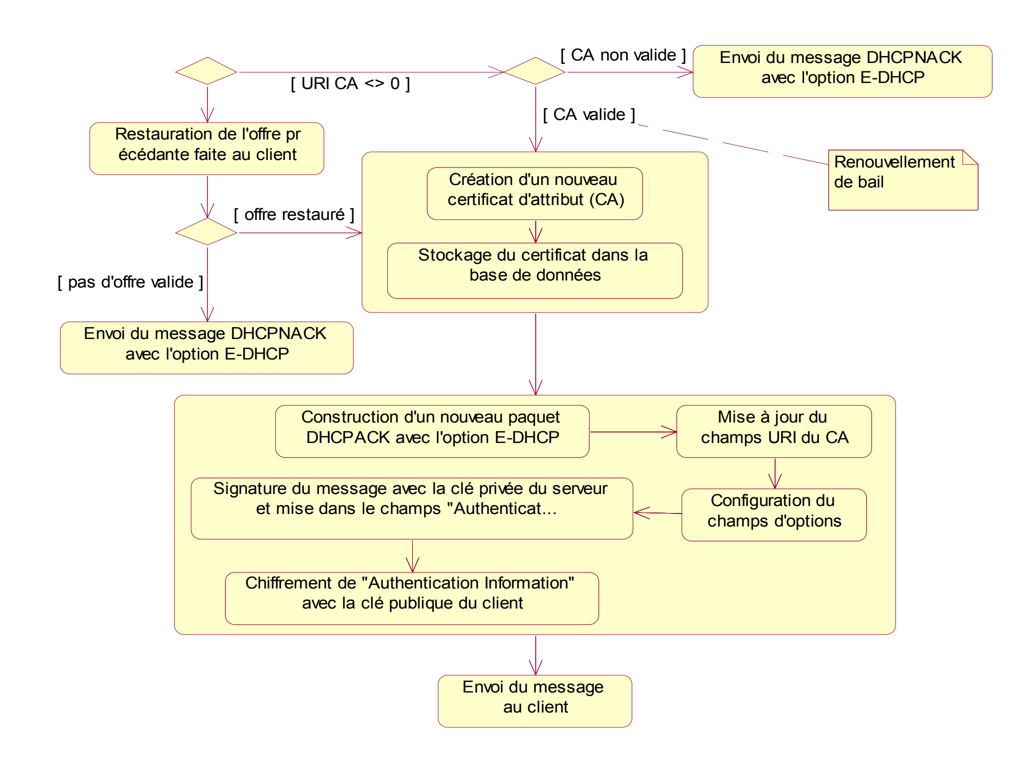


Figure 7-18 Diagramme d'états et transitions : Traitement du message *DHCPREQUEST*

➢ Diagramme d'états : « Traitement du message *DHCPINFORM* »

Après désencapsulation de l'option *E-DHCP* depuis un message *DHCPINFORM*, le serveur tente de restaurer la configuration requise pour le client. En cas d'échec, un message *DHCPNACK* authentifié avec *E-DHCP* est transmis au client (voir figure 7-19).

Sinon, la configuration est mise dans un message *DHCPACK* authentifié avec *E-DHCP* à destination du client.

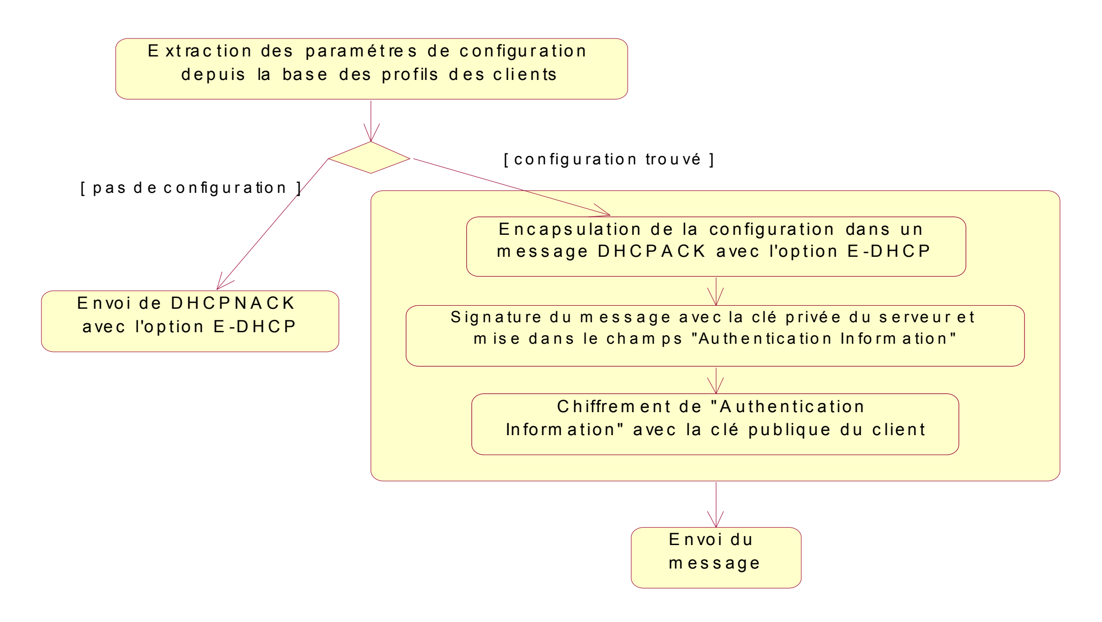


Figure 7-19 Diagramme d'états et transitions : Traitement du message *DHCPINFORM*

- **Diagramme d'états : « Traitement du message *DHCPRELEASE* »**

Après désencapsulation de l'option *E-DHCP* et selon la politique appliqué au serveur, celui-ci révoquera ou pas le certificat d'attributs référencé par le champ "*URIAttributeCertificate*" de l'option *E-DHCP*. Puis, le serveur rendra disponible l'adresse IP associée à ce certificat et transmettra, au client, un message *DHCPACK* authentifié avec *E-DHCP*. Dans le cas où la désencapsulation de l'option *E-DHCP* échoue, le serveur transmettra, au client, un message *DHCPNACK* authentifié avec *E-DHCP* (voir figure 7-20).

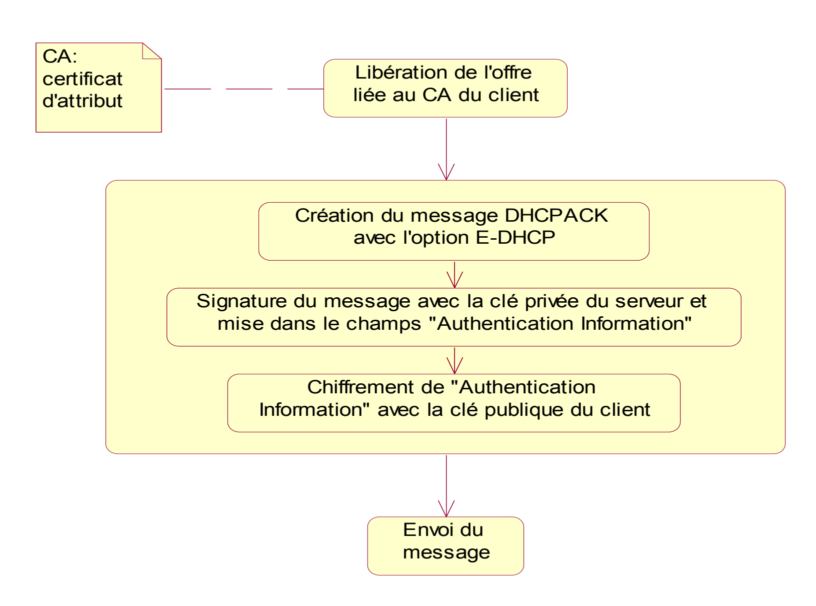


Figure 7-20 Diagramme d'états et transitions : Traitement du message *DHCPRELEASE*

- **Diagramme d'états : « Traitement du message *DHCPDECLINE* »**

Après désencapsulation de l'option *E-DHCP* depuis le message *DHCPDECLINE* reçu, le serveur libère l'offre faite précédemment au client, puis lui transmet un message *DHCPACK* authentifié avec *E-DHCP* (voir figure 7-21).

En cas d'échec de la désencapsulation, le serveur répond par un message *DHCPNACK* authentifié avec *E-DHCP*.

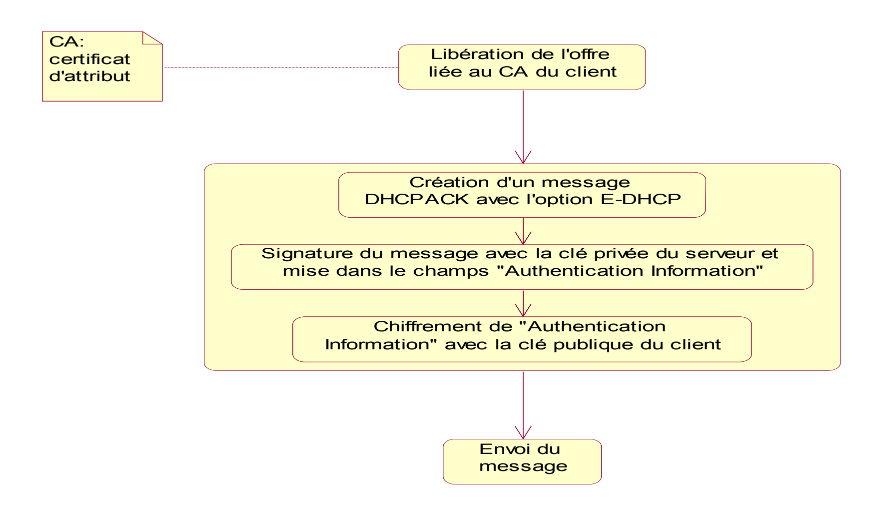


Figure 7-21 Diagramme d'états et transitions : Traitement du message *DHCPDECLINE*

7.1.4.2 Client

- **Diagramme d'activités du client *E-DHCP***

Après le démarrage, un client *E-DHCP* envoie un message *DHCPDISCOVER*. Tant que le client n'a pas encore d'adresse IP, il utilise l'adresse source *wildcard* ; 0.0.0.0 en Ipv4.

Le client précise dans le champ "*URIIdentityCertificate*" de l'option *E-DHCP* l'*URI* de son certificat X.509, met le champ "*URIAttributeCertificate*" à 0, puis signe le message *DHCP* et stocke la signature dans le champ "*AuthenticationInformation*" de l'option *E-DHCP*. Deux cas de figure se présentent (voir figure 7-22):

- le client ne connaît pas le serveur *E-DHCP*: avant de signer le message, le client positionne le champs "*Flag*" à 0, puis envoie le message en broadcast (adresse destination : 255.255.255.255, en IPv4).
- Le client connaît le serveur *E-DHCP* : avant de signer le message, le client positionne le champs "*Flag*" à 1, puis chiffre la signature avec la clé publique du serveur, stocke la signature chiffrée dans le champ "*AuthenticationInformation*" de l'option *E-DHCP* et envoie le message *DHCP* au serveur.

Le client attend le(s) offre(s) du/des serveur(s) *E-DHCP* et lorsqu'il en sélectionne une, vérifie l'authenticité du message, récupère le certificat X.509 du serveur référencé par le champ "*URIIdentityCertificate*" du message sélectionné, puis il transmet un message *DHCPREQUEST* authentifié avec l'option *E-DHCP*, avec le champ "*Flag*" mis à 1 (chiffrement de la signature avec la clé publique du serveur).

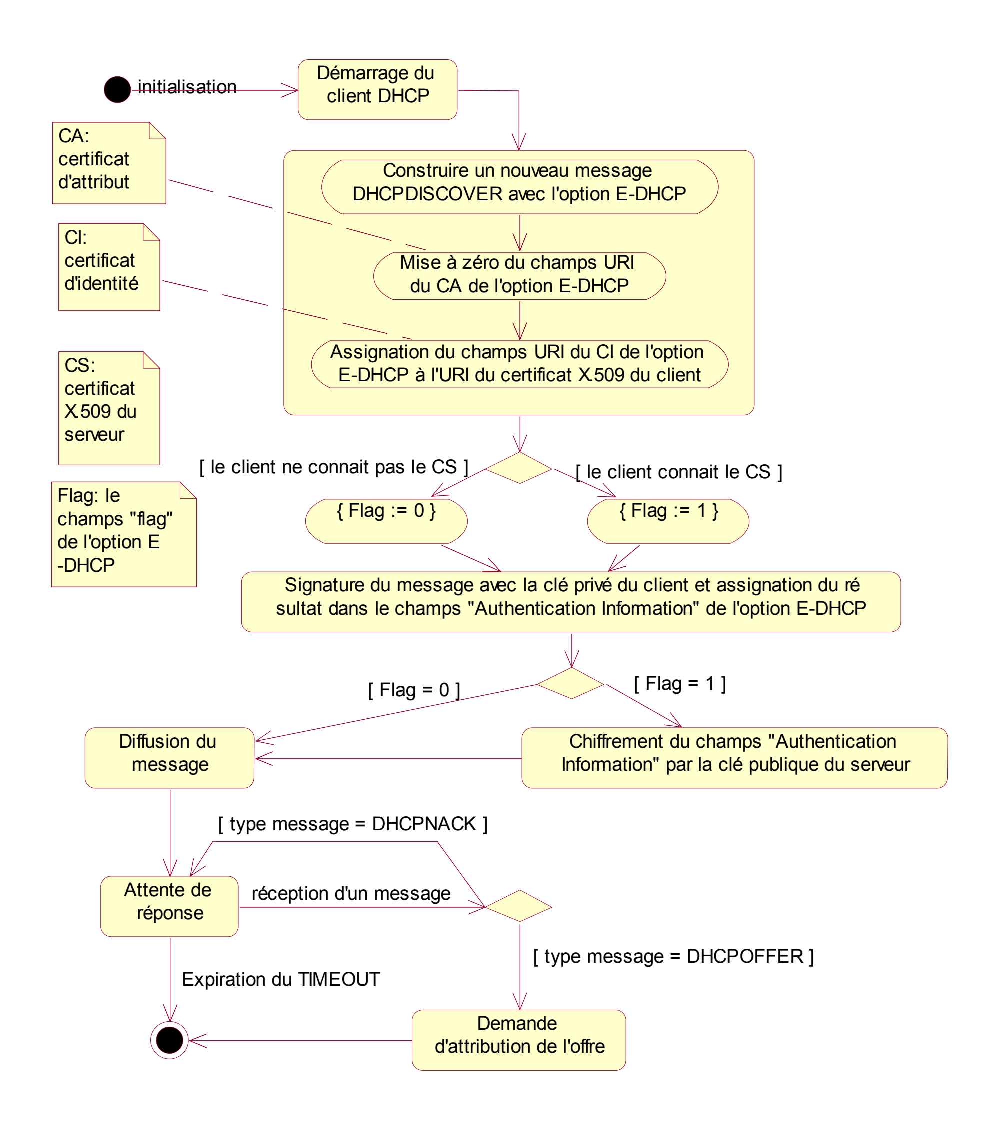


Figure 7-22 Diagramme d'activités du client *E-DHCP*

- **Diagramme d'états : « Demande d'attribution de l'offre »**

Lorsque le client *E-DHCP* accepte une offre faite par un serveur *E-DHCP*, il répond à ce serveur par un message *DHCPREQUEST*, ce message encapsule l'option d'authentification *E-DHCP*. Cette option a le champ "*Flag*" mis à 1 (chiffrement de la signature avec la clé publique du serveur *E-DHCP*). Le client attend la réponse du serveur, s'il reçoit un message *DHCPACK* authentifié avec *E-DHCP*, il désencapsule l'option *E-DHCP* puis se configure en fonction des paramètres extraits de ce message. Sinon, il recommence toute la procédure d'allocation d'adresse IP (voir figure 7-23).

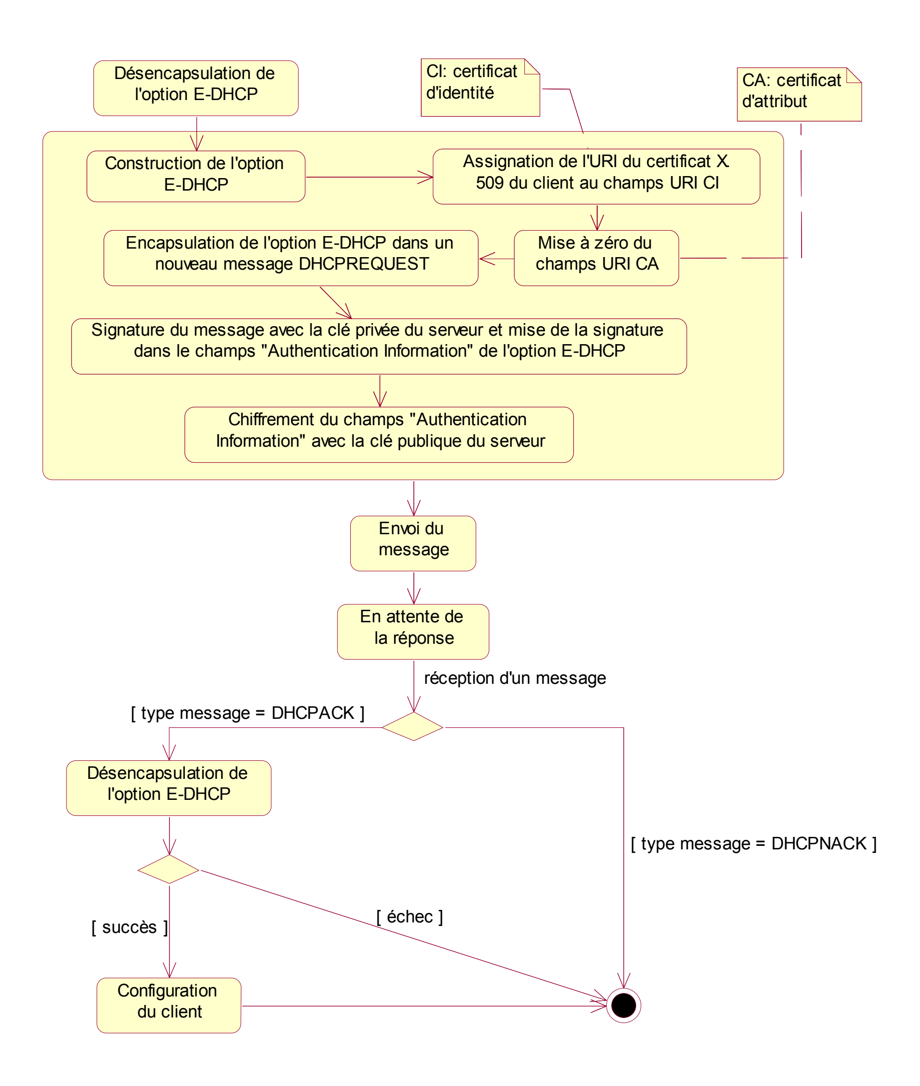


Figure 7-23 Diagramme d'états et transitions : Demande d'attribution de l'offre

7.2 Implémentation de *E-DHCP*

Après avoir présenté la modélisation en UML d'*E-DHCP* dans la partie précédente, nous nous intéréssons dans cette partie à son implémentation sur un client et sur un serveur *DHCP*.

Nous avons réutilisé le logiciel *DHCP* proposé par l'*ISC* (*Internet Software Consortium*) sous licence *GPL* (*General Public License*). Nous présentons donc notre examen du code source de ce logiciel. Enfin, nous expliquons comment nous avons modifié les sources d'*ISC DHCP*, afin de rajouter l'option proposée par *E-DHCP*.

7.2.1 *ISC DHCP* : étude du code source des logiciels client et serveur

Le logiciel *DHCP* de l'*ISC* [ISC] est une implantation du protocole *DHCP* qui comporte un ensemble d'outils (un serveur *DHCP*, un client *DHCP*, le support de *DHCP* pour les relais).

Pour le développement, seul le client et le serveur ont été utilisés. Le serveur relais ne devrait pas poser de problèmes particuliers dans le sens où il ne fait que transmettre les paquets sans prendre en compte le contenu des options.

Les sources de logiciel *DHCP* de l'*ISC* se présentent ainsi. Un répertoire *« common »* contient les fichiers de définition des fonctions communes au client et au serveur. Un répertoire *« client »* contient le code spécifique au client, tandis que le répertoire *« server »* contient le code source spécifique au serveur. « *relay* » contient les sources permettant de définir des serveurs relais. « *includes* » contient les fichiers de déclaration de fonctions (*fichier *.h*).

La compilation fait apparaître, du moins sur nos stations, un répertoire *./work.linux-2.2/* dans lequel se trouvent en particulier les exécutables.

Les attributions d'adresses IP sont stockées dans un fichier de « *leases* ». Ce fichier n'est pas créé par défaut, il est donc nécessaire de le rajouter pour permettre le fonctionnement du serveur *DHCP*. Le client demande un fichier de configuration, un fichier de *leases* "*dhclient.leases*", et un *script* de configuration correspondant à l'*OS* utilisé. Comme le client, le serveur *DHCP* réclame un fichier de *leases* "*dhcpd.leases*" (pour plus des détailles voir Annexe D).

7.2.1.1 Analyse du code commun au client et au serveur

Outre son répertoire spécifique, chacun des modules utilise le repertoire *includes/* qui contient les déclarations des structures et des fonctions ainsi que le répertoire *common* qui contient les fonctions partagées par le client et le srveur *DHCP*. Le répertoire *common/* contient quant à lui les fichiers de définition (*.c*) correspondants.

7.2.1.1.1 dhcp.h

Ce fichier contient les déclarations relatives à la forme d'un paquet *DHCP*. En particulier, il définit les différentes tailles des champs du paquet ainsi que la liste des options supportées par le logiciel. La structure *dhcp_packet* contient les différents champs *DHCP* (*op*, *htype*, *hlen*, etc.) sous forme « brute », d'entiers ou de tableaux d'entiers. En particulier, et cela est très utile, si la variable packet est de type *dhcp_packet**, alors un simple cast (*unsigned char *)* packet permet d'avoir un unique tableau d'entier qui est en fait le paquet.

Le fichier contient d'autres déclarations permettant de manipuler à l'intérieur du programme les différentes valeurs possibles pour les champs prenant leur valeur dans un ensemble de taille finie (type de matériel, drapeau, type du message...).

7.2.1.1.2 tree.h

Ce fichier contient les définitions d'un certain nombre de structure se présentant sous forme d'arbre dans le programme. Nous allons en présenter quelques-unes.

7.2.1.1.2.1 data_string et buffer

Ces deux structures sont fréquemment utilisées dans tout le code, y compris dans celui que nous avons rajouté. Elles se présentent ainsi :

```
struct buffer {
        int refcnt;
        unsigned char data [1];
};
/* A string of data bytes, possibly accompanied by a larger buffer. */
struct data_string {
        struct buffer *buffer;
        const unsigned char *data;
        unsigned len; /*Does not include NUL terminator, if any.*/
```

```
int terminated; };
```

Le champ data de *data_string* est défini comme constant, ce qui signifie que l'on est pas censé écrire dessus (d'ailleurs, un **(ds->data)='0'* dans le code provoque un *warning* lors de la compilation). Très souvent dans le code, on rencontre une initialisation de la forme *ds->data = ds->buffer->data*. On peut donc interpréter ce champ comme un moyen de lecture et le *buffer* comme un moyen d'écriture.

Plusieurs *data_string* peuvent pointer vers un même *buffer*, et c'est le rôle du champ *refcnt* de les comptabiliser. Les fonctions *buffer_reference* et *buffer_dereference* de *alloc.c* le gèrent pour nous.

7.2.1.1.2.2 expression

La lecture de sa déclaration et du enum *expr_op* qui la précède est aisée et s'assimile presque à une grammaire *BNF*. Elle est utile car c'est dans ce type de structure que sont stockées les informations de configuration après lecture du fichier de configuration. Or, les données pour *E-DHCP* se trouveront avant tout dans le fichier de configuration (*flag*, certificat d'identité, ...) ; il est donc nécessaire de les retrouver dans les multiples structures du programme.

7.2.1.1.3 dhcpd.h

Ce fichier contient l'essentiel des déclarations des structures et des fonctions. Cela est utile si l'on recherche le prototype d'une fonction, ou éventuellement dans quel fichier une fonction donnée est définie. Certaines des structures déclarées sont également très utiles :

7.2.1.1.3.1 option_cache

Cette structure sert à manipuler, tout au long du programme, les options présentes dans les paquets *DHCP*. Avec cette structure, une option peut être décrite soit par une expression, soit par une *data_string*. Par exemple, si oc est de type *option_cache**, on pourra trouver :

`oc -> data -> data` pour lire les données « brutes » de l'option (sans le code ni la longueur).

`oc -> data -> buffer -> data` pour écrire ces mêmes données.

`oc -> data -> len` pour connaître la longueur (sans le code ni la longueur, conformément à la RFC).

`oc -> option -> code` pour obtenir le code de l'option.

7.2.1.1.3.2 lease et lease_state

Ces structures permettent de manipuler les baux proposés par le serveur ou reçus par le client. *Lease* contient un champ *lease_state* qui, côté serveur, représente le traitement en cours du *bail* (si le champ est nul, le *bail* est libre).

7.2.1.1.4 alloc.c

Ce fichier contient des fonctions permettant d'allouer et de désallouer de la mémoire pour les structures en particulier *data_string* et *buffer*.

La fonction *buffer_allocate* permet de créer un *buffer* de taille len. Les fonctions *buffer_reference* et *buffer_dereference* permettent de gérer la valeur de refcnt dans la structure *buffer* suivant que le buffer est manipulé ou non par une structure.

La fonction *data_string_copy* permet de copier le contenu d'une *data_string* dans une autre. Il n'y a pas d'allocation de mémoire mais simplement modification des valeurs des pointeurs et mise à jour du compteur pour le *buffer*. La fonction *data_string_forget* modifie la valeur du refcnt du *buffer* qui est désallouée si plus aucun objet ne pointe dessus. Tous les pointeurs de la structure sont initialisés à *NUL*. La fonction *data_string_truncate* permet de tronquer la longueur de la chaîne de caractères manipulée par modification de la valeur de len dans la structure.

7.2.1.1.5 tables.c

Le fichier *tables.c* permet de définir le format des données contenues par une option, et ce pour plusieurs « universes ». Dans notre cas, on ne s'intéresse qu'au *dhcp_universe* et aux *dhcp_options*.

7.2.2 Implémentation de la solution *E-DHCP* dans ISC *DHCP*

Nous présentons dans cette partie les modifications que nous avons apportées au code pour intégrer l'option *E-DHCP*. Les fonctions nouvellement créées sont toutes regroupées dans la bibliothèque commune *edhcp* ; pour les clients et serveurs, nous présenterons toutes les fonctions modifiées.

7.2.2.1 Déclaration de l'option E-DHCP

Nous avons choisi d'attribuer à l'option *E-DHCP* le numéro 211, encore non utilisé. Dans *dhcp.h*, nous avons donc ajouté : `#define DHO_DHCP_E_DHCP 211`

Il fallait ensuite déclarer la structure d'une telle option dans *tables.c*. La documentation en tête de ce fichier est assez claire, et nous avons ajouté la ligne suivante dans les dhcp_options :

```
{ "e-dhcp", "fttX",  &dhcp_universe, 211 }
```

Cela signifie notemment que le parseur des fichiers de configuration s'attendra à trouver une ligne de la forme : `option e-dhcp flag ″chaine1″ ″chaine2″ data ;`

où `flag` vaut `on` ou `off`

et `data` est une donnée hexadécimale, par exemple 00:00:00:00

C'est ainsi que nous avons ajouté au fichier *dhclient.conf* la ligne :

```
send e-dhcp off "svr/certId/idclt" "" ff:ff:ff:ff ;
```

La donnée à la fin n'a aucun impact mais doit être présente. En effet, elle sera réallouée par la suite pour être de la bonne taille (cf la fonction *edhcp_expression_to_edhcp_option*). Pour le serveur, la ligne est la suivante :

```
option e-dhcp off "svr/certId/idsvr" "svr/certAttr/" ff:ff:ff:ff ;
```

Ici, on considère que la deuxième chaîne est en fait le répertoire où se trouveront les certificats d'attributs. Comme pour le client, la donnée hexadécimale n'a pas d'impact.

7.2.2.2 Makefiles

Puisque nous introduisons de nouveaux fichiers dans le code, il est nécessaire de modifier les *Makefile* en conséquence. Premièrement, nous avons créé une bibliothèque *E-DHCP*, dans le fichier *edhcp.c*. Il faut que ce fichier soit compilé et linké ; dans le *common/Makefile.dist*, nous avons donc ajouté *edhcp.c* (resp. *edhcp.o*) à la variable *SRC* (resp. *OBJ*).

7.2.2.3 La structure e_dhcp_option

La structure *e_dhcp_option* a été rajoutée dans dhcpd.h afin de stocker les informations relatives à l'option tant sur le serveur que sur le client. Elle est définie ainsi :

```
struct e_dhcp_option {
 unsigned char *flag ;            // auth_id chiffré par clef publique ?
   struct data_string *cert_id ;   // URI certificat d’identité
   struct data_string *cert_attr ; // URI certificat d’attributs
   struct data_string *auth_id ;   //  informations d'authentification };
```

La structure *e_dhcp_option* a été rajoutée dans la structure lease qui définit un *bail* attribué par le serveur à un client.

Elle est également utilisée dans la structure *client_state* qui définit l'état du client. Cette structure contient également le pointeur *private_key* vers la clef privée (*data_string*) du client afin de permettre la création du champ "*auth_id*".

7.2.2.4 La bibliothèque edhcp

Le fichier *edhcp.c* contient l'essentiel des fonctions permettant de manipuler l'option e-dhcp :

- La fonction *print_info()* affiche quelques informations sur la sortie standard. Elle permet de voir si les informations contenues dans le fichier de configuration sont correctement interprétées. Elle n'est pas vraiment utile pour le code lui-même, mais la lire permet de comprendre facilement où sont stockées les informations de configuration pour le client.
- La fonction *find_edhcp_expression* recherche dans une structure executable_statement une expression contenant l'option *E-DHCP*, si celle-ci est présente. Elle est utilisée par le client exclusivement, pour repérer l'option *E-DHCP* dans la configuration.
- La fonction *realloue* change la taille du *buffer* sur lequel pointe une data_string. Une nouvelle *data_string* contenant un buffer de la taille voulue est créée, puis celle-ci est copiée vers la *data_string* que l'on souhaite modifiée. L'ancien *buffer* est perdu et le contenu du nouveau est initialisé à 0.
- La fonction *desalloue_string_edhcp* désalloue les trois *data_string* de la structure *e_dhcp_option* passée en argument. Elle ne s'occupe absolument pas des *buffers* de ces data_string, ils peuvent même ne pas exister. Nous verrons l'utilité de cela avec la fonction *parse_edhcp_option*.
- La fonction *alloue_string_edhcp* alloue et initialise à 0 les trois structures data_string du edhcp en argument. Elle ne crée aucun buffer.
- La fonction *edhcp_expression_to_edhcp_option* permet le passage d'une expression contenant l'option *E-DHCP* à une structure *e_dhcp_option*. L'expression peut typiquement être obtenue par la fonction *find_edhcp_expression*. Concrètement, elle fait pointer *flag*, *cert_id* et *cert_attr* vers les *data_string* correspondantes de l'expression. La *data_string auth_id*, quant à elle, peut ne pas être à la bonne taille (car elle provient généralement de la lecture du fichier de configuration) ; elle est donc réallouée.
- La fonction *make_client_edhcp* initialise la structure *e_dhcp_option* pointée dans *client_state* après la lecture du fichier de configuration.
- La fonction *make_server_edhcp* crée et renvoie une *e_dhcp_option* à partir de la configuration lue côté serveur. La *e_dhcp_option* renvoyée (qui sera en *fait edhcp_cfg*, cf la partie serveur) est destinée à être utilisée par le serveur pour accéder aux champs de l'option qu'elle envoie, en particulier le *cert_attr*. Nous avons choisi de dire que le *cert_attr* défini dans la configuration est le répertoire vers les certificats d'attributs ; nous le copions donc dans la variable path donnée.
- La fonction *init_client_edhcp* initialise l'option côté client sur toutes les interfaces devant être configurées.
- La fonction *goto_next_zero* est utilisée par *parse_edhcp_option*.
- La fonction *parse_edhcp_option* parse le contenu (brut, fourni sous forme de data_string) correspondant à une option *e-dhcp* reçue dans un paquet. Elle stocke les résultats dans les champs d'une *e_dhcp_option*, mais *ne crée aucun buffer*. En effet, ces derniers servent plutôt à écrire des données, et la mémoire est désallouée en cas d'appel à *buffer_dereference*. On veut éviter cela car on pointe au milieu de la *data_string* décrivant l'option.
- La fonction *clear_auth_id* réinitialise la signature à 0 dans le paquet.

- La fonction *get_server_key* va rechercher la clef du serveur à l'endroit indiqué dans la réponse envoyée par le serveur.
- La fonction *check_signature* vérifie que la signature est valide, en calculant la signature attendue, et en vérifiant son égalité avec la signature effective.
- La fonction *is_valid_signature* fait appel à la précédente afin de vérifier que le champ "*Information d'Authentication*" a été correctement rempli.
- La fonction *cherche_edhcp* recherche l'option *e-dhcp* au sein d'un paquet fourni sous forme « brute », quasiment un tableau de caractères.
- La fonction *make_signature* crée la valeur d'authentification à partir du paquet formé avec une signature mise à zéro.
- La fonction *make_attribute_certificate* crée un certificat d'attributs pour le client et copie son nom (absolu) dans le champ *cert_attr* de l'*e_dhcp_option* passé en argument. Cette dernière est supposée provenir d'un *parse_edhcp_option()*, et n'a donc pas de *buffer*. Le *buffer* est donc créé et rempli avec le nom du certificat.
- La fonction *copie_cert_attr* copie (en allouant de la mémoire) le nom du certificat d'attributs d'une *option_cache* vers un *e_dhcp_option*. En pratique, l'*option_cache* proviendra d'un paquet.

7.2.2.5 Le client E-DHCP

Notons avant tout que nous avons ajouté un champ *edhcp*, de type *e_dhcp_option*, dans la structure *client_state*.

Dans la fonction main, le client commence par vérifier que les ressources adéquates sont disponibles, et vérifie qu'aucun client n'est en train de fonctionner. Il alloue les ressources nécessaires (port, etc...), puis lit le fichier de configuration. Juste après, nous avons ajouté une étape *E-DHCP* consistant à initialiser l'option en fonction des données présentes dans le fichier de configuration.

Le principe de fonctionnement du client est qu'une fonction examine les paquets entrants, et en fonction du type du paquet reçu, fait l'appel de la fonction appropriée (cette fonction a pour nom « *dhcpnomdutype* »).

7.2.2.5.1 Fonction dhcpoffer

Après avoir vérifié la présence des paramètres requis, le client inspecte la présence de l'option *E-DHCP*, qui peut être dans le paquet *E-DHCP* puisque le client a envoyé son premier paquet en utilisant cette option. Si elle existe, alors on continue. Sinon, on rejette le paquet sans réponse au serveur, et on loggue ce paquet.

On vérifie ensuite la signature d'authentification de l'option. Si elle n'est pas valide, on jette le paquet dans les mêmes conditions que précédemment. Sinon, on récupère l'URI du certificat d'attributs et on la stocke dans la structure *E-DHCP*. Ainsi, en cas de renouvellement ultérieur du bail, le client sera capable de rappeler au serveur en référence l'URI de ce certificat.

7.2.2.5.2 Fonction dhcpnack

Dans ce cas, le client vérifie que le message reçu est bien signé du serveur. En effet, si cette vérification n'était pas faite, un attaquant pourrait mettre fin prématurément à n'importe quelle négociation *DHCP* en envoyant au client un *dhcpnack* forgé. Si le message est correctement signé, alors le message est pris en compte classiquement. Sinon, le message est ignoré et l'événement est loggué.

7.2.2.5.3 Fonction dhcpack

Dans cette fonction aussi, par rapport au comportement traditionnel du client, nous avons rajouté une vérification de la signature du serveur. Si la signature n'est pas valide, le paquet est ignoré et l'événement est loggué.

7.2.2.5.4 Fonction make_discover

C'est dans les fonctions *make_** que sont fabriqués les paquets destinés à être envoyés. Ici, on examine si la structure *E-DHCP* du client existe (cela correspond au cas où le fichier de configuration demande l'activation de *E-DHCP*). Si c'est le cas, le client signe le paquet qu'il s'apprêtait à envoyer.

Concrètement, on remarque que cette fonction appelle *make_client_options()*. Nous n'avons pas modifié cette dernière, mais c'est elle qui ajoute les options dans le paquet, en particulier l'option *E-DHCP*.

Une fois que le paquet est entièrement construit, à la fin de la fonction, on appelle la fonction *make_signature()* de la bibliothèque *edhcp*, qui effectue la signature proprement dite.

7.2.2.5.5 Fonctions make_request, make_decline, make_release

Le comportement est analogue : en fonction de l'existence de la structure *E-DHCP*, on signe ou non le paquet.

7.2.2.6 Le serveur E-DHCP

La structure du serveur est beaucoup plus complexe que celle du client, nous nous sommes donc concentrés sur les points qui nous paraissaient les plus importants, sans entrer dans le détail du code. Nous avons modifié deux fichiers du serveur : *dhcpd.c* et *dhcp.c*.

7.2.2.6.1 Le fichier dhcpd.c

Il s'agit du fichier contenant la fonction *main()*. Pour autant, il ne contient pas le code le plus important du serveur.

7.2.2.6.1.1 Variables globales

Nous avons ajouté deux variables globales destinées à faciliter le traitement de l'option *E-DHCP* :

La première est *edhcp_cfg*, qui pointera vers les informations lues dans le fichier de configuration. Ainsi, en modifiant ses champs, on peut choisir les données qui seront incluses dans le paquet.

La deuxième est *path_cert_attr*, une *data_string* devra contenir le chemin vers les certificats d'attributs délivrés. Pour des raisons de simplicité, cette donnée sera copiée à partir des informations de configuration.

7.2.2.6.1.2 Fonction postconf_initialization()

Cette fonction est appelée depuis *main()* juste après la lecture du fichier de configuration. Nous y avons ajouté deux lignes, dont :

```
edhcp_cfg = make_server_edhcp (oc, &path_cert_attr) ;
```

qui initialise les deux variables mentionnées précédemment.

7.2.2.6.2 Le fichier dhcp.c

Ce fichier contient le code de fonctionnement du serveur. On pourra remarquer qu'il commence par une fonction *dhcp()*, dont le rôle est comparable à celui de la fonction *dhcp()* du client (c'est-à-dire

qu'elle appelle les fonctions *ad hoc* selon le type de paquet reçu), mais en effectuant plusieurs vérifications supplémentaires. Comme dans le cas du client, les fonctions appelées portent des noms explicites : *dhcpnomdutype*.

7.2.2.6.2.1 ***La fonction dhcpdiscover***

Si l'option *E-DHCP* est configurée, cette fonction commence par vérifier la signature du paquet reçu, puis intialise le *lease -> edhcp* avec l'option *E-DHCP* du paquet.

Cela est fait par un appel à *parse_edhcp_option*, ce qui signifie que les *data_string* construites ne sont pas « complètes » : elles n'ont pas de *buffer* car elles pointent au milieu du paquet. Manipuler de tels *buffer* comme dans le reste du programme serait désastreux car on pourrait désallouer de la mémoire appartenant au paquet !

Cette fonction se contente de manipuler des baux ; elle ne construit ni n'envoie le paquet réponse. Pour cela, elle appelle la fonction *ack_lease* que nous verrons plus tard. Au retour de cette fonction, nous détruisons la *lease -> edhcp*, et la lease considérée est déréférencée.

7.2.2.6.2.2 ***La fonction dhcprequest***

Cette fonction fait essentiellement la même chose que la précédente (induisant les mêmes commentaires), au détail près qu'elle crée un certificat d'attributs pour le client et stocke son *URI* dans *lease -> edhcp -> cert_attr*.

7.2.2.6.2.3 ***Les fonctions dhcprelease et dhcpdecline***

On se contente pour ces fonctions relativement simples de vérifier la présence de l'option *E-DHCP* et la signature fournie.

7.2.2.6.2.4 ***La fonction ack_lease***

Cette longue fonction est appelée en particulier par dhcpdiscover et dhcprequest. Elle construit une grande partie du paquet, notamment les options. Pour cela, elle utilise les informations établies lors de la lecture du fichier de configuration. C'est là qu'intervient notre variable globale *edhcp_cfg* : elle pointe vers les éléments adéquats dans la structure où sont enregistrées ces informations, et permet donc de les modifier à souhait.

Ici, on ne souhaite modifier que le certificat d'attributs : celui d'identité et le *flag* restent ceux de la configuration ; quant à la signature, elle sera traitée à part juste avant l'envoi effectif du paquet. La modification consiste donc à copier le champ *cert_attr* du *lease -> edhcp* passé en argument dans *edhcp_cfg -> cert_attr*.

ack_lease n'envoie pas le paquet elle-même : après avoir « ping-é » l'adresse du bail qu'elle s'apprête à accorder (et vérifié que personne ne répondait), elle appelle dhcpreply.

7.2.2.6.2.5 ***La fonction dhcpreply***

C'est elle qui envoie réellement le paquet (avec *send_packet*), c'est donc là que l'on réalise la signature grâce, comme d'habitude, à la fonction *make_signature*.

7.3 Conclusion

Dans ce chapitre, nous avons présenté la modélisation *UML* puis l'implémentation de *E-DHCP*. Cette modélisation nous a servi pour faciliter l'implémentation de *E-DHCP*. Nous avons présenté notre examen du code source du logiciel *DHCP* proposé par l'*ISC*. Ensuite, nous avons expliqué comment nous avons travaillé sur ces sources pour obtenir le résultat final de l'inplémentation d'*E-DHCP*.

Le travail que nous avons effectué nous a amenés à identifier quelques difficultés théoriques dans le protocole *E-DHCP*.

Ces difficultés sont liées, pour l'essentiel, à la protection du protocole contre le rejeu d'une transaction, et à la difficulté de transmettre les certificats d'identité des serveurs dans le cas fréquent où le client utilise *DHCP* pour obtenir sa première adresse réseau et n'a donc pas accès à des *URI* disponibles uniquement dans le réseau et non en local.

Nous avons cependant largement modifié les sources d'*ISC DHCP* afin de rajouter l'option proposée par *E-DHCP*. Tous les messages échangés sont désormais signés par les clients et les serveurs.

Pour assurer la protection du protocole contre le rejeu d'une transaction, on a requéris la présence du champ "*xid*" et aussi son incrémentation à chaque échange. On a exigé aux entités (client et serveur *E-DHCP*) de concatener à leur signature la valeur du champ "*xid*" afin de les chiffrer avec la clef publique du récepteur du message *DHCP*. Seulement le possesseur de la clef privée convenable (le récepteur) peut déchiffrer la valeur contenue dans le champ "*Information d'Authentication*" et ainsi peut savoir la valeur du champ "*xid*" (voir section 6.3.6.1).

Il reste donc à trouver un moyen pratique de transmettre le certificat d'identité du serveur lorsque le client n'a pas encore accès au réseau. En effet, dans les messages échangés, ce n'est pas le certificat lui-même qui est communiqué au client, mais l'*URI* permettant d'accéder à ce certificat.

Une solution pourrait être d'utiliser le protocole *TFTP* [RFC1350] de manière à demander une diffusion du certificat vers toutes les stations du sous-réseau. Toutefois, il faut signaler que le risque de collision est important si deux machines effectuent simultanément un transfert. De toute façon, le transfert du fichier ne sera pas sécurisé car *TFTP* ne prévoie pas de signer les messages de données échangés.

Chapitre 8

8 Conclusions générales et perspectives

8.1 Conclusions

La sécurité est un enjeu majeur des technologies numériques modernes. Avec le développement de l'Internet, les besoins de sécurité sont de plus en plus importants. Le développement d'applications Internet telles que le commerce électronique, les applications médicales ou la vidéoconférence, implique de nouveaux besoins comme, l'identification des entités communicantes, l'intégrité des messages échangés, la confidentialité de la transaction, l'authentification des entités, l'anonymat du propriétaire du certificat, l'habilitation des droits, la procuration, etc..
Qu'il s'agisse de données médicales, fiscales ou bancaires, le besoin en sécurité est essentiel afin de crédibiliser le système, tout en respectant à la fois les besoins des utilisateurs et des applications. Cette sécurité a néanmoins un prix : celui de l'établissement de la confiance entre les partenaires en communication. La confiance des utilisateurs passe par la sécurisation des transactions, par exemple au moyen d'une procédure de certification, et la reconnaissance des signatures électroniques.
Malgré la diversité des certificats numériques existants (certificat d'identité X.509, *SPKI*, certificat d'attributs, etc.), ils sont encore limités, génériques et répondent ainsi insuffisamment aux besoins spécifiques des applications électroniques et des utilisateurs. D'où la nécessité de spécifier une nouvelle approche pour la génération de certificats numériques répondant à ces exigences, légers, simplifiés et plus ouverts que ceux existants.
Cette thèse traite les services d'autorisation et leur intégration au protocole d'attribution dynamique des adresses *DHCP* (*Dynamic Host Configuration Protocol*). Elle est constituée de deux parties principales.

Dans la première partie, nous avons proposé et spécifié une approche qui permet de garantir à l'application et à l'utilisateur la bonne mise en forme des informations dans le certificat et l'adéquation du contenu de leur certificat vis-à-vis de leurs besoins. Ces certificats sont des certificats d'attributs spécifiés en *XML*, flexibles et respectant les besoins de l'application et la personnalisation de l'utilisateur durant la génération du certificat.
Pour chaque application, nous avons défini une grammaire *DTD* (*Document Type Definition*) pour préciser tous les champs dont l'application a besoin. L'idée principale est de stocker, sur le serveur, des *DTDs* c'est-à-dire, des fichiers contenant un certain nombre de paramètres correspondant aux données qui seront insérées dans le certificat d'attributs final. La génération de ces certificats d'attributs respecte la grammaire associée à l'application. En effet, c'est grâce à celles-ci que l'administrateur personnalisera les certificats d'attributs que l'utilisateur pourra demander. Ainsi, si le besoin d'un nouveau type de certificat d'attributs émane, il suffit de créer la *DTD* correspondant à la nouvelle application ajoutée.
Pour satisfaire les besoins de l'utilisateur, l'*E-IGP* (Extension de l'Infrastructure de Gestion des Privilèges) permet à ce dernier de personnaliser sa demande de certificats d'attributs. C'est l'utilisateur qui précise les valeurs des paramètres de l'application, la date de validité de son certificat d'attributs, les rôles qu'il souhaite avoir ou les délégations qu'il souhaite fournir à quelqu'un suivant ces besoins.
La mise en œuvre d'*E-IGP* a nécessité l'existence d'une Infrastructure de Gestion des Clefs auquelle l'*E-IGP* est rattaché.

Nous signalons que les résultats de cette première partie de notre thèse ont été publiés dans plusieurs conférences, [SETIT04], [SAR03], [SETIT03], [SSGRR03], [SAR02].

Dans la deuxième partie de notre thèse, nous avons traité la problématique de l'introduction de la sécurité dans le protocole d'attribution dynamique des adresses Internet : *DHCPv4*.
Le protocole d'attribution dynamique des adresses Internet est nécessaire pour le fonctionnement d'un nombre considérable de réseaux et cela pour entre autres, deux raisons. La première raison est le manque d'adresses Internet qui ne permet pas leur attribution de manière statique. La deuxième raison est que la mobilité des équipements reste plus adaptée à l'adressage dynamique.
Ce protocole est aujourd'hui très répandu, car il s'avère très pratique pour gérer de larges parcs de machines. Son usage est tout à fait indiqué dans les entreprises et les universités, tant pour affecter aux machines des adresses IP statiques que des adresses IP dynamiques. Il permet de centraliser complètement la gestion des ressources en adressage.
Si son succès est indiscutable, il faut avouer que la sécurité n'était pas une préoccupation majeure à l'époque où il a été mis au point, en 1993, puis amélioré, en 1997. En fait, il n'y avait aucune tentative dans la conception de *DHCP* pour mettre en place des systèmes de protection contre les hôtes d'Internet malveillant. Par conséquent le protocole est vulnérable à une variété d'attaques. Les problèmes de sécurité de *DHCP* viennent du fait qu'il n'y a aucun mécanisme d'authentification des entités *DHCP* (client/serveur) et des messages *DHCP*.
Plusieurs extensions visant à parer à ses principales failles de sécurité intrinsèques ont été proposées. Pour autant, aucune d'entre elles ne corrige tous les problèmes!
Pour résoudre ces problèmes, nous avons proposé une extension au protocole *DHCP*, appelée *E-DHCP* (*Extended-Dynamic Host Configuration Protocol*) qui permet un contrôle strict sur des équipements par une procédure d'authentification forte. Cette extension consiste à rajouter une nouvelle option *DHCP* qui fournit simultanément l'authentification d'entités et de messages *DHCP* en se basant sur l'utilisation de clefs de chiffrement asymétrique RSA, de certificats d'identité X.509 et de certificats d'attributs (proposés dans la première partie). Cette extension de *DHCP* est possible car le champ d'options prévoit l'implémentation de nouvelles options.
De plus, nous avons adossé le serveur *E-DHCP* à un serveur d'attribution de privilèges *AA* d'un *E-IGP*. Le serveur *E-DHCP* crée un certificat d'attributs pour le client contenant l'adresse Internet attribuée dynamiquement. Pour leur authentification au sein des architectures réseaux, les équipements peuvent faire preuve de l'authenticité de leur adresse en présentant leur certificat d'identification X.509 et le certificat d'attributs. Ce certificat d'attributs sera ensuite utilisé dans le contrôle d'accès aux services.
Notre proposition *E-DHCP* a été validée par une implantation. Après la modélisation d'*E-DHCP*, nous avons largement modifié le code libre "*open source*" de *DHCP*, développé par le ISC (*Internet Software Consortium*) afin de rajouter l'option proposée par *E-DHCP*. Nous avons par la suite, développé une autorité d'attribution (*AA*) que nous avons intégrée au serveur *E-DHCP*.

Nous signalons que les résultats de cette deuxième partie de notre thèse ont été publiés dans plusieurs conférences [ICETE], [SEC], [ICTTA], [SAR04].

Enfin, il sera bien de noter que les résultats des travaux réalisés dans le cadre de notre thèse concernant l'extension du protocole *DHCP* ont été repris par l'ENST (Ecole Nationale Supérieure des Télécommunications) et France Télécom R&D pour une contribution à la normalisation au sein de l'*IETF*. Cette contribution a été publiée sous forme de draft [DEMERDraft].

8.2 Perspectives

Les perspectives d'évolution de ce travail sont multiples. Sur le plan théorique, la technique d'authentification utilisé dans *E-DHCP* [DEMERDraft] fonctionne sans aucun problème avec le

protocole *DHCP* dans sa version six. Sur le plan technique, une évolution intéressante consisterait à implémenter *E-DHCP* au protocole *DHCPv6* [RFC3315]. Pendant notre thèse nous n'avons pas eu le temps de faire cette implémentation. Sa serait très intéressant de valider l'interopérabilité d'*E-DHCP* avec *DHCPv6* par une implantation à échelle réelle.

Un autre axe d'étude important est le sujet de l'interopérabilité entre *IPSec* [RFC2401] et *DHCPv4*. En effet, le protocole *IPSec* repose intrinsèquement sur les adresses qui permettent d'appliquer la politique de sécurité. On ne peut pas donc permettre que les adresses IP puissent changer dynamiquement et indépendamment d'*IPSec*. Les protocoles d'allocation dynamique, tels qu'ils sont utilisés aujourd'hui, ne permettent pas de gérer simplement des politiques de sécurité basées sur des adresses IP. L'utilisation conjointe de l'allocation dynamique et d'*IPSec* est donc très limitative. Nous pensons que la solution (*E-DHCP*) que nous proposons s'articule parfaitement avec le protocole *IPsec* par l'utilisation des certificats d'attributs crée par *E-DHCP*. Nous signalons que le protocole de gestion des clefs *ISAKMP* supporte les certificats d'attributs. Dans une future direction de notre recherche, nous comptons valider l'interopérabilité de notre proposition avec *IPSec* par une implantation à échelle réelle.

Le sujet de l'incompatibilité entre *IPSec* avec les fonctions de translations d'adresses IP (*NAT*) [RFC2663] d'une part et les fonctions d'assignation dynamique d'adresses (*DHCP*) d'une autre part nous s'avère un axe d'étude très important. Le principe de mise en oeuvre d'*IPSEC* est la définition commune, au niveau de l'équipement source et de l'équipement destination, d'une association permettant de définir les flux à chiffrer notamment à partir des adresses IP, les algorithmes cryptographiques à utiliser et bien évidemment les clefs de chiffrement (mode *ESP*) ou d'identification (mode *AH*). Ce principe relativement simple à la base, peut s'avérer difficile à mettre en oeuvre. En effet, les traitements réalisés au niveau des paquets IP (ajout d'information dans le datagramme, masquage des entêtes en mode tunnel) peuvent s'avérer incompatibles avec les fonctions de translations d'adresses IP (*NAT*), d'assignation dynamique d'adresses (*DHCP*) ainsi qu'avec d'éventuelles règles de fragmentation des paquets IP. Sa serait très intéressant de travailler sur ce sujet dans un futur travail.

Liste des publications

1.	J. Demerjian, A. Serhrouchni, M. Achemlal. "Extended DHCP". IETF Draft. Date d'expiration février 2005. Link: http://www.ietf.org/internet-drafts/draft-demerjian-serhrouchni-achmelal-edhcp-00.txt.
2.	J. Demerjian, A. Serhrouchni, M. Achemlal. "Certificate-based access control and authentication for DHCP". ACM/IEEE International Conference on E-business and Telecommunication Networks ICETE'2004, INSTICC Press 2004, ISBN 972-8865-15-5, pp.99, Setubal, Portugal, 25-28 août 2004.
3.	J. Demerjian, A. Serhrouchni. "DHCP authentication using certificates". 19th IFIP International Information Security Conference SEC'2004, ISBN 1-4020-8142-1, pp.457, Centre de Congrés Pierre Baudis – Toulouse, France, 23-26 août 2004.
4.	J. Demerjian, M. Achemlal. "Extension du protocole DHCP pour l'attribution dynamique des adresses avec authentification forte". 3ème Conférence sur la Sécurité et Architecture Réseaux SAR'04, La Londe, Côte d'Azur, France, 21-25 juin 2004.
5.	J. Demerjian, A. Serhrouchni. "E-DHCP : Extended Dynamic Host Configuration Protocol". IEEE 1st International Conference on Information & Communication Technologies : From Theory to Applications ICTTA'04, ISBN 0-7803-8484-2, pp.667, Damascus, Syria, 19-23 avril 2004.
6.	J. Demerjian, A. Serhrouchni. "EPMI : Une extension de l'infrastructure de gestion des privilèges". IEEE International Conference Sciences of Electronic, Technology of Information and Telecommunications SETIT'2004, ISBN 9973-41-902-2, pp.205, Sousse, Tunisie, 15-20 mars 2004.
7.	J. Demerjian, A. Serhrouchni, F. Tastet. "La certification sous un nouvel angle de vue". 2ème Conférence sur la Sécurité et Architecture Réseaux SAR'03, pp.135, Nancy, France, 30 juin- 4 juillet 2003.
8.	J. Demerjian, A. Serhrouchni, F. Tastet. "Une nouvelle approche pour la certification". IEEE International Conference Sciences of Electronic, Technology of Information and Telecommunications SETIT'2003, ISBN 9973-41-685-6, pp.59, Sousse, Tunisie, 17-21 mars 2003.
9.	J. Demerjian, A. Serhrouchni, F. Tastet. "Why certificates don't meet e-Business needs". Fifth International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, on the Internet SSGRR 2003w, ISBN 88-85280-75-7, pp.58, L'Aquila, Italie, 6-12 janvier 2003.
10.	J. Demerjian, A. Serhrouchni. "XML et Sécurité". 1ère Conférence sur la Sécurité et Architecture Réseaux SAR'02, Marrakech, Maroc, 8-14 juillet 2002.

Références bibliographiques

[BAL] L'infrastructure PKI de Baltimore Technologies. Web Site: http://www.baltimore.com/.

[BUL] A. Buldas, P. Laud, H Lipmaa and J. Villemson. Time-Stamping with Binary Linking Schemes. Advances in Cryptology, vol. 1462, pp. 486-501, CRYPTO'98.

[CBDA] G. Glazer, C. Hussey and R. Shea. Certificate-Based Authentication for DHCP [Electronic Version]. Retrieved from UCLA university, Computer Science department Web Site: http://www.cs.ucla.edu/~chussey/proj/dhcp_cert/cbda.pdf , March 2003.

[CERT02] Cert^{R} Advisory CA-2002-12 format String Vulnerability in ISC DHCPD. Retrieved from: http://www.cert.org/advisories/CA-2002-12.html, May 2002.

[CERT03] Cert^{R} Advisory CA-2003-01 Buffer Overflows in ISC DHCPD Minires Library. Retrieved from: http://www.cert.org/advisories/CA-2003-01.html, January 2003.

[CGI] The CGI Specification. Retrieved from: http://hoohoo.ncsa.uiuc.edu/cgi/interface.html.

[COH] E Cohen. A Short Course on Computer Viruses. Wiley and Sons, 1994.

[DIF] W. Diffie and M. E. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory, Vol.IT-22, pp. 644-654, November 1976.

[DNAC] B. Zouari, H. Afifi et A. Mhamed. "Mobilité et sécurité dans les réseaux Wi-Fi". 16ème Congrès DNAC : De Nouvelles Architectures pour les Communications. Paris, France, 2-4 décembre 2002.

[DUNL] M. Lai. UML; La notation unifiée de modélisation objet Objet; Applications en java. DUNODE 2ème édition.

[DUNF] A.-F. Fausse. La signature électronique. DUNOD. 2001.

[ETSI] European Telecommunications Standards Institute. Retrieved from: http://www.etsi.org/.

[EYRAUT] T. Autret, L. Bellefin et M.-L. Oble-Laffaire. Sécuriser ses échanges électroniques avec une PKI. EYROLLES. 2002.

[EYRJAN] C. Jan et G. Sabatier. La sécurité informatique. EYROLLES 1989.

[EYRPUJ] Guy Pujolle. Les Réseaux. EYROLLES. 3ème édition, 2000.

[GIPMDS] Groupement d'Intérêt Public Modernisation des Déclarations Sociales. Lien: http://www.gip-mds.fr/.

[GRES] H. ElBakkali et B. Kaitouni. L'analyse formelle du modèle de confiance d'une PKI à l'aide d'une logique de croyance. "GRES'01, Gestion de REseau et de Service". Marrakech, Maroc, 17-21 Décembre 2001.

[GuiUML] James Rumbaugh et Ivar Jacobson. Le guide de l'utilisateur UML. EYROLLES.

[HAB] S. Haber and W.S. Stornetta. How to Time Stamp a Digital Document. Journal of Cryptology, vol. 3, N°2, pp. 99-111, 1991.

[HAY] Hayes, J.M. The Problem with Multiple Roots in Web Browsers – Certificate Masquerading. In IEEE Computer Society Proceedings of WETICE 1998, 17-19 June 1998 at Palo Alto, California.

[HELS] L. Sundström. The Dynamic Host Configuration Protocol (DHCP). Helsinki University of Technology, Department of Computer Science. Retrieved from: http://www.cs.hut.fi/~ljs/dhcp/dhcp.pdf.

[HSC] G. Labouret. PKI et Certificats. Hervé Schauer Consultants (HSC). Retrieved from: http://www.hsc.fr/ressources/presentations/pki/img0.htm.

[IALTA] Groupe de Travail Certificats d'Attributs. Lien: http://www.ialtafrance.org/content/association/certificats_attributs_html.

[ICARE] Projet ICARE (Infrastructure de Confiance sur des Architectures de Réseaux Internet & Mobile). Lien: http://www.cert-i-care.org/ICare_documents.htm.

[ICETE] J. Demerjian, A. Serhrouchni & Mohammed Achemlal. Certificate-based Access Control and Authentication for DHCP. "ACM/IEEE ICETE 2004, International Conference on E-Business and Telecommunication Networks". Setúbal, Portugal, 25-28 August 2004.

[ICTTA] J. Demerjian & A. Serhrouchni. E-DHCP: Extended Dynamic Host Configuration Protocol. "IEEE ICTTA'04, IEEE 1st International Conference on Information & Communication Technologies: From Theory to Applications". Damascus, Syria, 19-23 April 2004.

[IETF] Internet Engineering Task Force. Web site: http://www.ietf.org/

[INET] A. Tominaga, O. Nakamura, F Taraoka and J. Murai. Problems and Solutions of DHCP. The 5th Annual Conference of the Internet Society. INET'95, 1995.

[IPv6] G. Fernandes, S. Léon et V. Victor. Le protocole Internet version 6 (IPv6). Security-labs. Lien : http://www.security-labs.org/index.php3?page=105. Mai 2004.

[ISC] ISC: Internet Software Consortium. Dynamic Host Configuration Protocol Distribution. Retrieved from: http://www.isc.org/index.pl?/sw/dhcp/, February 2004.

[ISO] International Organisation for Standardisation. Web site: http://www.iso.org/iso/en/ISOOnline.frontpage

[ISS] The ISS (Internet Security Systems) company. Novell NetWare malformed DHCP request denial of service. Retrieved from: http://xforce.iss.net/xforce/xfdb/9428, June 2002.

[IUT2004] Sécurité dans les télécommunications et les technologies de l'information. Aperçu des problèmes et présentation des Recommandations UIT-T existantes sur la sécurité dans les télécommunications. Lien: http://www.itu.int/itudoc/itu-t/85097-fr.pdf. Décembre

2003.

[JAV] Products & Technologies Java Technology. Web Site: http://java.sun.com/

[LCN] T. Komori and T. Saito. The secure DHCP System with User Authentication. IEEE the 27th Annual IEEE Conference on Local Computer Networks, LCN'02, pp 0123, 2002.

[LCS] MIT Laboratory for Computer Science. Web Site: http://www.lcs.mit.edu/.

[MacNN] MacNN (Mac News Network). Malicious DHCP response on OS X can grant root access. Retrieved from: http://www.macnn.com/news/22234&startNumber=10, November 2003.

[MISC02] MISC (Multi-System & Internet Security Cookbook) Magazine N°4. DHCP: spoof & destroy! . November December 2002.

[MITSDSI] R. Rivest and B.Lampson. SDSI – A Simple Distributed Security Infrastructure. Retrieved from: http://theory.lcs.mit.edu/~rivest/sdsi10.html, September 1996.

[O'RECac] C. Cachat et D. Carella. PKI Open Source. O'REILLY. Edition originale, 2003.

[O'REGiz] G. Gizault. IPv6 Théorie et pratique. O'REILLY. 3ème édition, 2002.

[OCTALIS] J. Ribes and X. orri. ASN.1 vs XML. Octalis Paper. Retrieved from: http://www.octalis.com/pdfs/rd/asn1vsXml.pdf, 15 January 2002.

[PAS] K. Jensen and N. Wirth. PASCAL - User Manual and Report. ISO Pascal Standard. Retrieved from: http://www.inf.ethz.ch/~wirth/books/Pascal/, Springer Verlag 1991.

[RatRose] Logiciel de modélisation UML. Rational Rose édition. Lien : www.ibm.com / www.rational.com

[RIT] D. M. Ritchie. C Reference Manual. Bell Telephone Laboratories. Murray Hill, New Jersey 07974. Retrieved from: http://cm.bell-labs.com/cm/cs/who/dmr/cman.pdf, May 1975.

[SAN] D. Sanai. Detection of promiscuous Nodes Using ARP packets. A white paper from http://www.securityfriday.com. August 2002.

[SAR02] J. Demerjian & A. Serhrouchni. XML et Sécurité. "SAR'02, 1st Conference on Security and Network Architectures ". Marrakech, Maroc, July 2002.

[SAR03] J. Demerjian, A. Serhrouchni, F. Tastet. La certification sous un nouvel angle de vue. "2ème Conférence sur la Sécurité et Architecture Réseaux SAR'03", pp.135, Nancy, France, 30 juin- 4 juillet 2003.

[SAR04] J. Demerjian, M. Achemlal. Extension du protocole DHCP pour l'attribution dynamique des adresses avec authentification forte. "3ème Conférence sur la Sécurité et Architecture Réseaux SAR'04", La Londe, Côte d'Azur, France, 21-25 juin 2004.

[SEC] J. Demerjian & A. Serhrouchni. DHCP authentication using certificates. "IFIP SEC 2004, 19th IFIP International Information Security Conference". Toulouse, France, 23-26 August 2004.

[SETIT03] J. Demerjian, A. Serhrouchni & F. Tastet. Une nouvelle approche pour la certification. "IEEE SETIT 2003, International Conference Sciences of Electronic, Technology of Information and Telecommunications". Sousse, Tunisie, pp.59, 17-21 March 2003.

[SETIT04] J. Demerjian & A. Serhrouchni. EPMI : Une extension de l'infrastructure de gestion des privilèges. "IEEE SETIT 2004, International Conference Sciences of Electronic, Technology of Information and Telecommunications". Sousse, Tunisie, ISBN 9973-41-902-2, pp.205, 15-20 March 2004.

[SEXP] Ron Rivest, code and description of S-expressions. Retrieved from: http://theory.lcs.mit.edu/~rivest/sexp.html.

[SIC] J. Virchaux. DHCP, la location d'adresses IP. SIC Flash Informatique, 18 September 2001, No. 7.

[SOAP] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1", Retrieved from: http://www.w3.org/TR/2000/NOTE-SOAP-20000508, May 2000.

[SSGRR03] J. Demerjian, A. Serhrouchni & F. Tastet. Why certificates don't meet e-Business needs. "SSGRR 2003w, Fifth International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, on the Internet". L'Aquila, Italie, pp.58, January 2003.

[SVM] E. Barret. DHCP, ou l'adressage IP facile. Revue SVM Mac, April 2002, No. 138.

[SYNIAL] Conclusions du groupe de Travail sur la gestion des Attributs (GT-GA). La qualité professionnelle dans la signature électronique. Lien : http://www.signelec.com/content/download/Synthese_GT_GA_v1_10.pdf, février 2007.

[THOSCH] B. Schneier. Cryptographie Appliquée. International THOMSON Publishing France. 2ème édition, 1997.

[TUTOR] B. Augustin & R. Le Guen. Usurpation d'identité sur Ethernet. Projet TUTORE. Université Paris XII – IUT de Créteil Vitry. Lien: http://www.romainl.com/down/Usurpation.htm.

[W3C] World Wide Web Consortium. Retrieved from: http://www.w3c.org/ .

[W3CDTD] Guide to the W3C XML Specification ("XMLspec") DTD, Version 2.1. Retrieved from: http://www.w3.org/XML/1998/06/xmlspec-report.

[WirNet] C. Perkins and K. Luo. Using DHCP with computers that move. The Wireless Networks, Vol. 1, No. 3, pp 341-354, 1995.

[XENC] T. Imamura, B. Dillaway and E. Simon. XML Encryption Syntax and Processing. W3C Recommendation. Retrieved from: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/, December 2002.

[XKMS] W. Ford, P. Hallam-Baker, B. Fox, B. Dillaway, B. LaMacchia, J. Epstein and J. Lapp. XML Key Management Specification (XKMS). Retrieved from: http://www.w3.org/TR/2001/NOTE-xkms-20010330/, March 2001.

[XSIG] D. Eastlake, J.Reagle and D. Solo. XML-Signature Syntax and Processing. W3C Recommendation. Retrieved from: http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/, February 2002.

Les recommandations de l'ITU-T

[ITUT] International Telecommunication Union – Telecommunication Standardization Sector. Web site: http://www.itu.int/ITU-T/.

[ITUT-X500-01] ITU-T Recommendation X.500. Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services.

[ITUT-X509-97] ITU-T Recommendation X.509. Information technology – Open Systems Interconnection – The Directory: Authentication frameworks, 1997.

[ITUT-X509-00] ITU-T Recommendation X.509. Information technology – Open Systems Interconnection – The Directory: Public-Key and attribute certificate framework, 2000.

[ITUT-X208-88] ITU-T Recommendation X.208. Specification of Abstract Syntax Notation One (ASN.1), 1988.

[ITUT-X209-88] ITU-T Recommendation X.209. Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), 1988.

[ITUT-X690-97] ITU-T Recommendation X.690. Information technologiy - ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER), 2002.

Les RFC de l'IETF

[RFC768] IETF RFC 768. User Datagram Protocol. August 1980.

[RFC791] IETF RFC 791. Internet Protocol. September 1981.

[RFC792] IETF RFC 792. Internet Control Message Protocol. September 1981.

[RRFC793] IETF RFC 793. Transmission Control Protocol. September 1981.

[RFC826] IETF RFC 826. Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. November 1982.

[RFC903] IETF RFC 903. Reverse Address Resolution Protocol. June 1984.

[RFC951] IETF RFC 951. Bootstrap Protocol. September 1985.

[RFC959] IETF RFC 959. File Transfer Protocol. October 1985.

[RFC1114] IETF RFC 1114. Privacy enhancement for Internet electronic mail: Part II - certificate-based key management. August 1989.

[RFC1122] IETF RFC 1122. Requirements for Internet Hosts- Communication Layers. October 1989.

[RFC1279] IETF RFC 1279. X.500 and Domains, November 1991.

[RFC1305] IETF RFC 1305. Network Time Protocol (Version 3) Specification, Implementation. March 1992.

[RFC1321] IETF RFC 1321. The MD5 Message-Digest Algorithm. April 1992.

[RFC1350] IETF RFC 1350. The TFTP Protocol (Revision 2). July 1992.

[RFC1459] IETF RFC 1459. Internet Relay Chat Protocol. May 1993.

[RFC1510] IETF RFC 1510. The Kerberos Network Authentication Service (V5). September 1993.

[RFC1534] IETF RFC 1534. Interoperation Between DHCP and BOOTP. October 1993.

[RFC1591] IETF RFC 1591. Domaine Name System Structure and Delegation. March 1994.

[RFC1738] IETF RFC 1738. Uniform Ressource Locators (URL). December 1994.

[RFC1767] IETF RFC 1767. MIME Encapsulation of EDI objects. March 1995.

[RFC1866] IETF RFC 1866. Hypertext Markup Language – 2.0. November 1995.

[RFC1991] IETF RFC 1991. PGP Message Exchange Formats. August 1996.

[RFC2104] IETF RFC 2104. HMAC: Keyed-Hashing for Message Authentication. February 1997.

[RFC2131] IETF RFC 2131. Dynamic Host Configuration Protocol. March 1997.

[RFC2132] IETF RFC 2132. DHCP Options and BOOTP Vendor Extensions. March 1997.

[RFC2162] IETF RFC 2162. MaXIM-11 – Mapping between X.400/ Internet mail and Mail-11 mail. January 1998.

[RFC2202] IETF RFC 2202. Test Cases for HMAC-MD5 and HMAC-SHA-1. September 1997.

[RFC2246] IETF RFC 2246. The TLS Protocol Version 1.0. January 1999

[RFC2251] IETF RFC 2251. Lightweight Directory Access Protocol (v3). December 1997.

[RFC2315] IETF RFC 2315. PKCS #7: Cryptographic Message Syntax Version 1.5. March 1998.

[RFC2396] IETF RFC 2396. Uniform Ressource Identifiers (URI). August 1998.

[RFC2401] IETF RFC 2401. Security Architecture for the Internet Protocol. November 1998.

[RFC2408] IETF RFC 2408. Internet Security Association and Key Management Protocol

(ISAKMP). November 1998.

[RFC2459] IETF RFC 2459. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. January 1999.

[RFC2460] IETF RFC 2460. Internet Protocol, Version 6 (IPv6) Specification. December 1998.

[RFC2462] IETF RFC 2462. IPv6 Stateless Address Autoconfiguration. December 1998.

[RFC2463] IETF RFC 2463. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. December 1998.

[RFC2485] IETF RFC 2485. DHCP Option for The Group's User Authentication Protocol. January 1999.

[RFC2489] IETF RFC 2489. Procedure for Defining New DHCP Options. January 1999.

[RFC2510] IETF RFC 2510. Internet X.509 Public Key Infrastructure Certificate Management Protocols. March 1999.

[RFC2511] IETF RFC 2511. Internet X.509 Certificate Request Message Format. March 1999.

[RFC2560] IETF RFC 2560. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP. June 1999.

[RFC2616] IETF RFC 2616. HyperText Transfer Protocol – HTTP/1.1. June 1999.

[RFC2663] IETF RFC 2663. IP Network Address Translator (NAT) Terminology and Considerations. August 1999.

[RFC2692] IETF RFC 2692. SPKI Requirements. September 1999.

[RFC2693] IETF RFC 2693. SPKI Certificate Theory. September 1999.

[RFC2807] IETF RFC 2807. XML Signature Requirements. July 2000.

[RFC2821] IETF RFC 2821. Simple Mail Transfer Protocol. April 2001.

[RFC2903] IETF RFC 2903. Generic AAA Architecture. August 2000.

[RFC2939] IETF RFC 2939. Procedure and IANA Guidelines for Definition of New DHCP Options and Message Types. September 2000.

[RFC2986] IETF RFC 2986. PKCS #10: Certification request Syntax Specification Version 1.7. November 2000.

[RFC3046] IETF RFC 3046. DHCP Relay Agent Information Option. January 2001.

[RFC3118] IETF RFC 3118. Authentication for DHCP Messages. June 2001.

[RFC3275] IETF RFC 3275. (Extensible Markup Language) XML-Signature Syntax and Processing. March 2002.

[RFC3279] IETF RFC 3279. Algorithms and Identifiers for the Internet X.509 Public Key Infarstructure Certificate and Certificate Revocation List (CRL). April 2002.

[RFC3280] IETF RFC 3280. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. April 2002.

[RFC3281] IETF RFC 3281. An Internet Attribute Certificate Profile for Authorization. April 2002.

[RFC3315] IETF RFC 3315. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). July 2003.

[RFC3396] IETF RFC 3396. Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4). November 2002.

[RFC3447] IETF RFC 3447. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. February 2003.

[RFC3456] IETF RFC 3456. Dynamic Host Configuration Protocol (DHCPv4) Configuration of IPSec Tunnel Mode. January 2003.

[RFC3470] IETF RFC 3470. Guidelines for the Use of extensible Markup Language (XML) within IETF Protocols. January 2003.

[RFC3645] IETF RFC 3645. Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG). October 2003.

[RFC3647] IETF RFC 3647. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. November 2003.

[RFC3845] IETF RFC 3845. DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format. August 2004.

[RFC3851] IETF RFC 3851. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. July 2004.

Les Darfts de l'IETF

[DEMERDraft] J. Demerjian, A. Serhrouchni, M. Achemlal. Extended DHCP. IETF Draft. Will Expire on February 2005. Retrieved from: http://www.ietf.org/internet-drafts/draft-demerjian-serhrouchni-achmelal-edhcp-00.txt.

[HOMDraft] IETF Internet Draft. Homstein and al. DHCP Authentication via Kerberos V, Retrieved from: http://quimby.gnus.org/internet-drafts/draft-hornstein-dhc-kerbauth-00.txt. Expire May 2000.

[MALDraft] IETF Internet Draft. Malpani, R. Housley, T. Freeman, Simple Certificate Validation Protocol (SCVP), draft-ietf-pkix-scvp-12, June 2003.

[ORRDraft] IETF Internet Draft. X. Orri and J. Mas. SPKI-XML Certificate Structure. Expire May 2002.

[PAADraft] IETF Internet Draft. J. Paarjarvi. XML Encoding of SPKI Certificates. March 2000.

[SENDDraft] IETF Internet Draft. J. Arkko and al. SEcure Neighbor Discovery. Retrieved from: ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-ietf-send-ndopt-06.txt. Expire January 2005.

[CGADraft] IETF Internet Draft. T. Aura. Cryptographically Generated Addresses (CGA). Retrieved from: ftp://ftp.rfc-editor.org/in-notes/internet-drafts/draft-ietf-send-cga-06.txt. Expire October 2004.

[ABKDraft] IETF Internet Draft. J. Kempf. Securing IPv6 Neighbor Discovery Using Address Based Keys (ABKs). Expire November 2003.

[AAADraft] IETF Internet Draft. C. Perkins and al. AAA for IPv6 Network Access. Expire November 2003.

Annexes

Annexe A

A.1 Notions de base de la cryptographie

La cryptologie est la science du secret : vieille de plusieurs millénaires, elle propose des méthodes de chiffrement pour assurer la confidentialité des données. Aujourd'hui, elle permet également d'assurer l'intégrité de l'information en vérifiant l'authenticité de son contenu et de sa provenance.

En tant que science du secret, elle se décompose en deux secteurs. Il s'agit d'une part de concevoir de nouveaux algorithmes permettant le chiffrement de données informatiques devant transiter par un canal non protégé : c'est le domaine de la *cryptographie*. La cryptographie, du grec Kruptos, caché, et graphein, écrire, est la science des écritures secrètes. La cryptographie assure des services de sécurité.

A l'inverse, il est aussi important d'évaluer et de mettre à l'épreuve les algorithmes existant de nos jours pour cerner leur degré de résistance à divers types d'attaques : c'est le domaine de la *cryptanalyse*. La cryptographie est donc indissociable de la cryptanalyse. Les utilisateurs autorisés, c'est-à-dire ceux qui possèdent les secrets, font du chiffrement ou du déchiffrement, alors que les cryptanalystes, les agresseurs, font du décryptage.

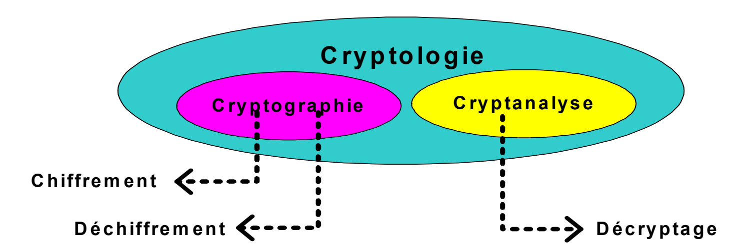


Figure 8-1 Cryptologie

Il existe deux catégories de procédés cryptographiques : les algorithmes à clef secrète et ceux à clef publique.

A.1.A Cryptographie symétrique

Les algorithmes à clef secrète sont dits *symétriques.* On parle de cryptographie symétrique lorsque les clefs d'encryptage et de décryptage sont les mêmes. Ceci implique que les deux entités en communications se sont au préalable mis d'accord sur la clef à utiliser dans leur échange d'informations.

L'avantage de ce type de systèmes réside dans ses performances et dans sa relative simplicité qui permet de l'embarquer sur des systèmes de petite taille ou limités en ressources.

Le cryptage à clef secrète souffre par contre de deux faiblesses qui empêchent son utilisation sur le réseau. La première réside dans le fait que la clef doit être remise aux deux participants par un autre canal, sûr, préalablement à toute communication sécurisée sur le réseau. L'échange de clefs dans un tel système est un problème non trivial.

La seconde faiblesse est due au nombre de clefs nécessaires, qui augmente avec le nombre n de participants au réseau sécurisé selon la formule : $n \div 2 \times (n-1)$.

Ce système est peu pratique dans le cas d'application de type grand public, comme le commerce électronique. Le destinataire de l'information aimerait que n'importe qui puisse l'atteindre, mais s'il publie sa clef à tous ses clients, celle-ci ne sera plus secrète, et la cryptographie devient inutile.

A.1.B Cryptographie asymétrique

De même, les algorithmes à clef publique sont dits *asymétriques* car ils utilisent deux clefs différentes : une clef publique et une clef privée.

La cryptographie à clef publique repose sur une idée simple : au lieu de l'échange d'une seule clef secrète, chaque usager possède une paire de clefs mathématiquement liées, différentes mais complémentaires. La première, la clef privée est gardée par son propriétaire, alors que la seconde, la clef publique, est largement diffusée par exemple sur un annuaire électronique. Le principe de cette méthode de cryptage est une fonction mathématique à sens unique, *one way function*, basée sur la propriété des nombres premiers.

La cryptographie à clef publique permet d'une part l'échange de la clef publique par l'intermédiaire d'un réseau non sécurisé et, d'autre part, ne nécessite qu'une seule clef par participant : à n participants d'un réseau sécurisé correspondent n clefs de chiffrement. La cryptographie asymétrique n'a pas le problème de transmission de clef secrète comme celle de la cryptographie symétrique.

La cryptographie asymétrique introduit un nouveau problème : l'utilisation d'une couple de clefs entraîne la nécessité de publication, en toute confiance, de la clef publique. Cette publication doit offrir l'assurance que :

· la clef est bien celle appartenant à la personne avec qui les échanges sont envisagés ;

· le possesseur de cette clef est « digne de confiance » ;

· la clef est toujours valide.

Cette notion de confiance est résolue avec les certificats et les autorités de certification.

Une analogie éclairante du fonctionnement du chiffrement asymétrique revient à considérer la clef publique comme un cadenas ouvert à disposition de tous les participants sur le réseau, et la clef privée à la clef de ce cadenas. Sécuriser un texte en clair à destination d'un participant donné sur le réseau consiste à se procurer la clef publique de ce participant par l'intermédiaire d'un tiers de confiance, un organisme certifié de distribution de clefs publiques, *Public Key Infrastructure* (PKI), et de chiffrer le texte en clair « en fermant le cadenas ». Seul ce participant à qui le cryptogramme est destiné sera capable de le déchiffrer puisque lui seul possède la clef du cadenas.

Les systèmes asymétriques permettent de rendre de nombreux services de sécurité, et en particulier ceux d'authentification et de non-répudiation.

A.2 Signature numérique

Le principe de la signature numérique consiste à appliquer une fonction mathématique sur une portion du message. Cette fonction mathématique s'appelle fonction de hachage et le résultat de cette fonction est appelé code de hachage. Ce code fait usage d'emprunte digitale du message. Il faut noter que la fonction est choisie de telle manière qu'il soit impossible de changer le contenu du message sans altérer le code de hachage.

Ce code de hachage est ensuite crypté avec la clef privée de l'émetteur et rajouté au message. Lorsque le destinataire reçoit le message, il décrypte ce code grâce à la clef publique de la source puis il compare ce code à un autre code qu'il calcule grâce au message reçu. Si les deux correspondent, le destinataire sait que le message n'a pas été altéré et que son intégrité n'a pas été compromise. Le destinataire sait aussi que le message provient de l'émetteur puisque seul ce dernier possède la clef privée qui a crypté le code.

Ce principe de signature fut amélioré avec la mise en place de certificats permettant de garantir la validité de la clef publique fournie par l'émetteur.

Dans toute transaction professionnelle, les deux parties doivent offrir une garantie de leur identité. La signature numérique et le certificat sont des moyens d'identification de l'émetteur du message.

A.3 Les quatre principes de base de la cryptographie

La cryptographie assure des services de sécurité. Ces services de sécurité présentés dans cette section vont s'appuyer sur différents mécanismes de sécurité qui pourront être utilisés seuls ou en combinaison (voir figure 11-10).

Les mécanismes de sécurité vont être appliqués aux données traitées par les applications et combinées à des informations de sécurité, dont les clefs cryptographiques.

A.3.A Authentification

Le service d'authentification d'un partenaire est la vérification des déclaratifs d'identité d'une personne ou d'un processus informatique. L'authentification a pour objectif de réduire sinon d'éliminer les risques de travestissement d'intrus sous d'apparences légitimes en vue de poursuivre des opérations non autorisées.

Dans le monde électronique, le service d'authentification des partenaires en communication va se faire par le contrôle des conventions privées de celui qui va se faire authentifier.

A.3.B Confidentialité

Le service de confidentialité a trait à la protection contre la consultation non autorisée de données stockées ou échangées. Pour les données stockées, ce service peut être rendu soit par le contrôle d'accès aux environnements de stockage, soit par le chiffrement des données stockées. Dans le cadre de données échangées, seul le mécanisme de chiffrement est possible.

Préalablement à l'échange, l'émetteur et le récepteur ont convenu d'un mot de passe commun connu uniquement d'eux. L'émetteur utilise ce mot de passe comme une clef de chiffrement pour chiffrer le message, et seul le bon récepteur qui connaît ce mot de passe peut l'utiliser comme clef de déchiffrement. Dans la réalité, des systèmes complexes de synchronisation des clefs sont utilisés par les ordinateurs des deux participants afin de rendre le système à la fois sûr et simple à utiliser.

Dans toute transaction professionnelle, les deux parties doivent offrir une garantie de leur identité. La signature numérique et le certificat sont des moyens d'identification de l'émetteur du message.

A.3.C Intégrité

Les services d'intégrité des réseaux visent à assurer le bon fonctionnement des ressources du réseau, et la transmission ou l'enregistrement sans problème des données sur le réseau. Ces services assurent une protection contre la modification délibérée ou accidentelle et non autorisée des fonctions du réseau (intégrité du système) et de l'information (intégrité des données).

A.3.C.1 Intégrité du système

Les mécanismes d'intégrité physique sont habituellement des solutions à faible coût et faciles à mettre en œuvre, et ils assurent l'intégrité physique d'un réseau, lorsque l'accès physique à celui-ci peut présenter un problème. Ces mécanismes peuvent être de natures diverses - mécanismes enregistreurs de violation, mécanismes inviolables, mécanismes avertisseurs de violation.

A.3.C.2 Intégrité des données

Les services d'intégrité des données (ou des messages échangés) aident à protéger les données (ou les messages émis) et les logiciels qui résident sur les postes de travail client, les serveurs de fichiers et les autres composants du réseau, contre les modifications non autorisées, lesquelles pouvant être de

nature intentionnelle ou accidentelle. Ce type de service peut être assuré par l'utilisation de valeurs de contrôle cryptographique[1], ainsi que des mécanismes très granulaires de privilège et de contrôle de l'accès. Plus le contrôle de l'accès ou le mécanisme de privilège est granulaire, moins une modification non autorisée ou accidentelle des données risque de se produire.

À la différence du contrôle des accès, les services d'intégrité des données permettent un moyen de vérifier si les données ont été altérées, supprimées ou ajoutées de quelque manière que ce soit pendant leur transmission.

A.3.D *Non-répudiation*

Par non-répudiation, on entend le service de sécurité qui permet aux entités qui participent à une communication de ne pas nier y avoir participé. Ce service comporte la production, l'accumulation, la recherche et l'interprétation de données à l'effet qu'une partie donnée a bel et bien traité un élément donné. Lorsque l'on utilise des services de non-répudiation, une entité qui, par exemple, a envoyé un message ne peut pas nier l'avoir fait (on parle alors de non-répudiation grâce à la preuve d'expédition) et l'entité qui l'a reçu ne peut pas nier l'avoir reçu (non-répudiation grâce à la preuve de réception). Ce service est particulièrement important dans le courrier électronique et dans les applications commerciales électroniques (EAA). Pour employer les services de non-répudiation, on doit utiliser des techniques cryptographiques avec clefs publiques et privées, ainsi que les signatures numériques. Lorsque les signatures numériques sont employées à cette fin, il est crucial que les clefs privées soient protégées contre la copie et la divulgation. On ne peut pas assurer la confiance envers les services de non-répudiation si on ne protège pas, au préalable, les clefs privées de signature.

Mécanisme / **Service**	Chiffrement	Signature numérique	Contrôle d'accès	Intégrité des données	Echange d'authentification	Bourrage	Contrôle du routage	Notarisation
Authentification de l'entité homologue	X	X			X			
Authentification des données d'origine	X	X						
Service de contrôle d'accès			X					
Confidentialité mode connecté	X						X	
Confidentialité mode non connecté	X						X	
Confidentialité d'un champ	X							
Secret du flux	X					X	X	
Intégrité mode connecté (avec récupération)	X			X				
Intégrité mode connecté (sans récupération)	X			X				
Intégrité mode connecté (d'un champ)	X			X				
Intégrité mode non connecté	X	X		X				
Intégrité mode non connecté (d'un champ)	X	X		X				
Non-Répudiation de l'origine		X		X				X
Non-Répudiation de la délivrance		X		X				X

Figure 8-2 Croisement entre services et mécanismes de sécurité

[1] Pour désigner les valeurs de contrôle cryptographiques, on utilise d'autres termes, notamment : code d'authentification des messages, contrôle de redondance cyclique chiffré et sceaux.

Annexe B

B.1 Modèle de confiance

Au-delà du déploiement des certificats, il est nécessaire de considérer leur niveau de confiance et ainsi toutes les vérifications et révocations nécessaires pour leur usage. Nous allons dans cette section traiter les aspects relatifs à leur usage.

B.1.A Certification hierarchique

La première méthode, qui est la plus simple, consiste en une hiérarchie d'autorités de certification que l'on peut représenter par un arbre. À la racine de cet arbre se trouve l'autorité racine. Aux nœuds intermédiaires, on trouve les différentes autorités de certification, et les feuilles correspondent aux utilisateurs finaux. Dans cet arbre, le fait que certains utilisateurs héritent d'une autorité de certification montre la relation de confiance qu'il existe entre eux. Les utilisateurs Alice et Bob, par exemple, font confiance à une autorité A, ce qui sous-entend que leur certificat de clef publique a été signé avec la clef publique de A. La clef publique de A sera quant à elle certifiée par une autorité de certification supérieure (La recommandation X.509 prévoit dans ses extensions la possibilité de signer un certificat d'utilisateur ou un certificat d'autorité de certification). On remonte ainsi l'arbre jusqu'à la racine. Ainsi, si Alice veut entrer en communication avec Cécile qui dépend de l'autorité de certification B, Alice va récupérer le certificat de Cécile pour initier la communication. Ce certificat a été signé par B. Soit Alice possède la clef publique de B et tout se passe normalement, soit elle ne la possède pas, et elle va devoir l'obtenir. Dans ce système, toutes les feuilles doivent posséder une copie de la clef publique de la racine. En partant de la racine, nous allons donc être capable, en vérifiant les certificats successifs et en interrogeant les différentes autorités de certification d'obtenir le certificat de B. A partir de ce moment, on remonte la chaîne des certificats jusqu'à la racine de façon à être capable d'extraire la clef publique de B.

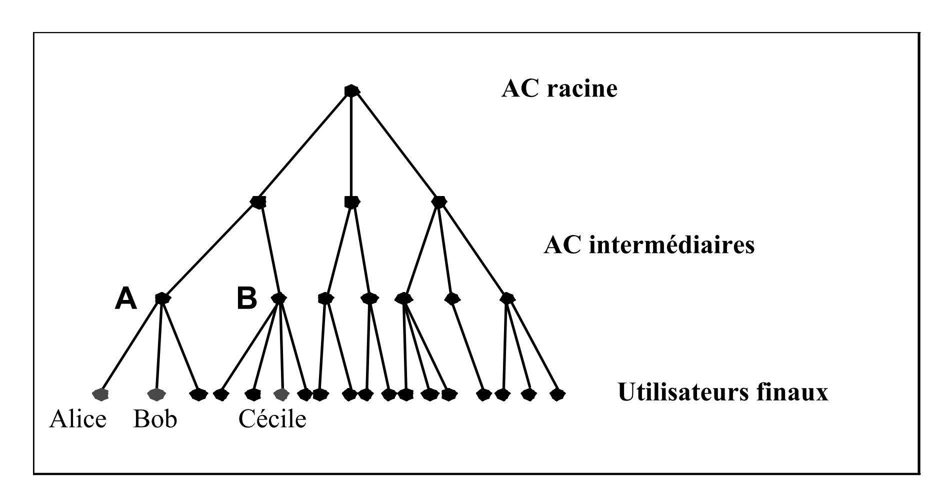


Figure 8-3 Hiérarchie simple des autorités de certification

L'avantage de cette méthode est sa simplicité. On s'aperçoit par contre que dans le cas où il existe plusieurs niveaux intermédiaires d'autorités de certification, la recherche de l'autorité et le chemin des certificats seront difficiles à obtenir.

B.1.B Certification mutuelle ou *cross certification*

La seconde méthode dérive de la première. On va donc limiter le modèle hiérarchique à un seul étage, ce qui est nettement suffisant dans la plupart des cas pratiques. Par contre, on va permettre aux racines des différents arbres de se faire confiance mutuellement. Pour cela, on met en œuvre la certification croisée. Lors de cette opération, l'autorité A va effectuer une certification de la clef publique de B (certification croisée unilatérale). Si l'opération est effectuée dans les deux sens, on dit que la certification croisée est mutuelle. De cette façon, l'ensemble des utilisateurs de la hiérarchie de A par exemple seront capables d'accéder aux utilisateurs de la hiérarchie de B, si A et B ont effectué une certification croisée. Il y a donc différentes solutions pour relier les différentes racines des différents domaines, la solution la plus simple étant de les lier de proche en proche, et la solution la plus complexe étant d'effectuer un maillage complet entre ces entités.

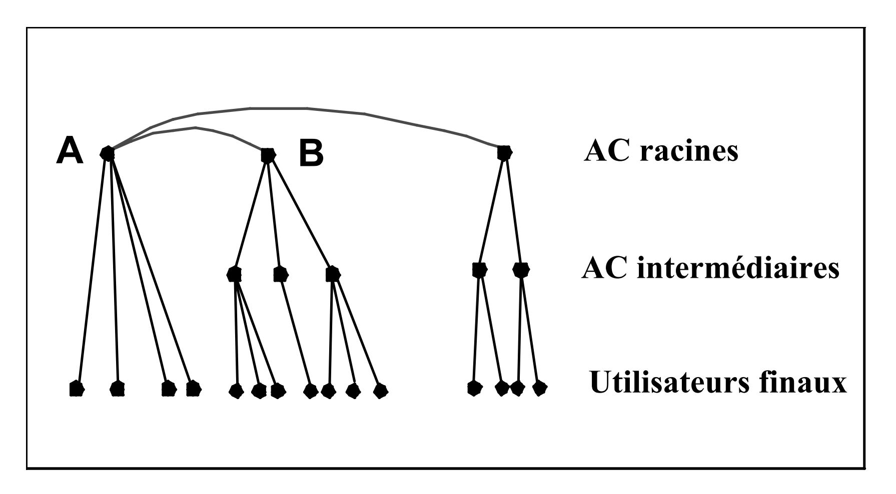


Figure 8-4 Modèle distribué

B.1.C La certification par acceptation préalable de l'usager

La troisième méthode concerne la certification sur Internet, cette méthode peut être éventuellement utilisée dans le cas des Intranets. Dans le cas d'Internet, les certificats sont généralement utilisés pour protéger les clefs publiques/privées nécessaires à l'établissement d'une session SSL ou encore au téléchargement de fichiers exécutables (ActiveX Plug-ins, tec…). Des autorités de certification éditent donc un certain nombre de certificats pour le compte d'entreprises implantées sur Internet. Pour qu'un client (ici un navigateur) fasse confiance à un site, celui-ci va lui présenter son certificat avant d'initier une connexion sécurisée. L'utilisateur va donc pouvoir accepter ou non le certificat (suivant le paramétrage du navigateur, ceci est soumis à l'approbation de l'utilisateur ou non). Simplement, il est impératif que le navigateur soit en possession des clefs publiques des autorités de certification desquelles il souhaite pouvoir lire les certificats. Ces clefs sont donc préinstallées dans le navigateur. Un même navigateur est donc une feuille de plusieurs arbres appartenant à une autorité de certification. Ce système, même s'il faut reconnaître qu'il est pratique, présente un certain nombre de problèmes de sécurité. Avant tout l'utilisateur d'un navigateur n'est pas forcément conscient des certificats auxquels il va faire confiance si son navigateur est paramétré de façon à faire confiance automatiquement aux certificats issus d'une autorité de certification installée par défaut. De plus, si une autorité de certification voit sa clef révoquée pour une raison ou une autre, les navigateurs restent toujours capables d'accepter des certificats de sa provenance à moins que l'utilisateur ne désactive manuellement (et pour cela, il faut qu'il soit prévenu, et qu'il soit capable de le faire) la clef publique de l'autorité de certification révoquée.

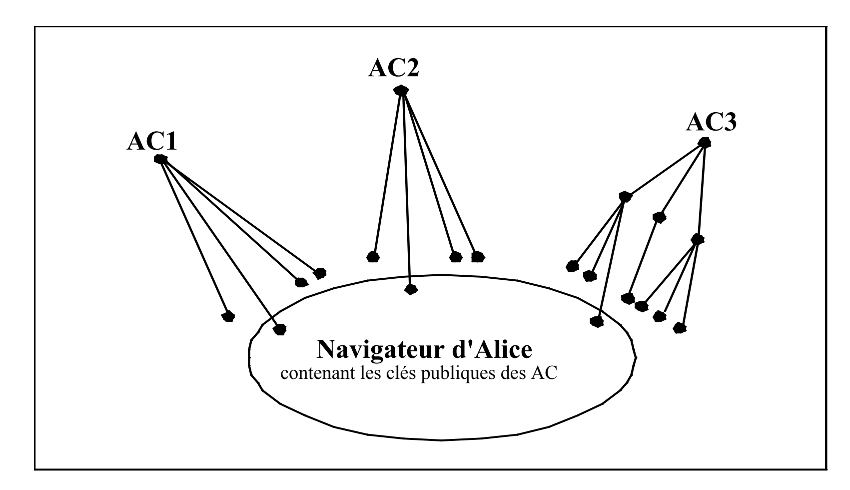


Figure 8-5 Modèle hiérarchique dans le cas du Web

B.1.D Le Web trust ou la confiance par connaissance

Le dernier modèle est utilisé par PGP (Pretty Good Privacy). Chaque utilisateur est lui-même une autorité de certification et décide quels sont les certificats sur lesquels il peut baser sa confiance et ceux qu'il doit rejeter. Grâce à la notion de transitivité, les membres d'une même communauté se font mutuellement confiance. Cette démarche, si elle fonctionne et est implantée, ne peut être étendue à des communautés importantes, puisque la recherche d'une chaîne de confiance ressemble plus à un tâtonnement dans le graphe des relations de confiance qu'autre chose. De plus, cette méthode n'est pas adaptée à une entreprise où l'utilisation de différentes clefs permet de cantonner les informations à certains secteurs.

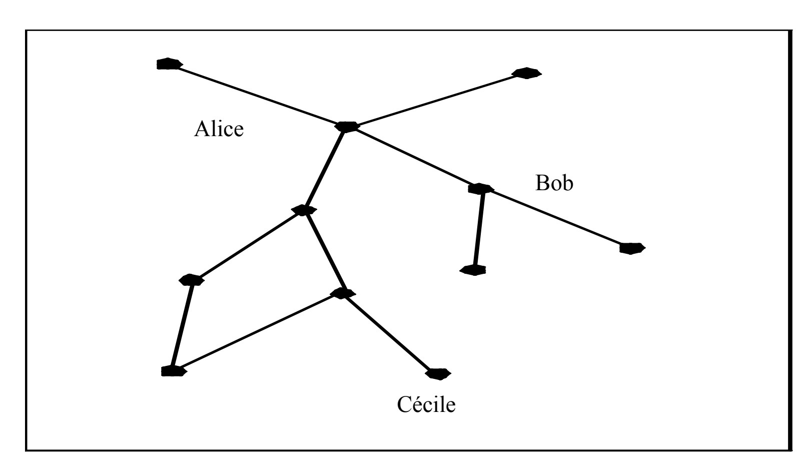
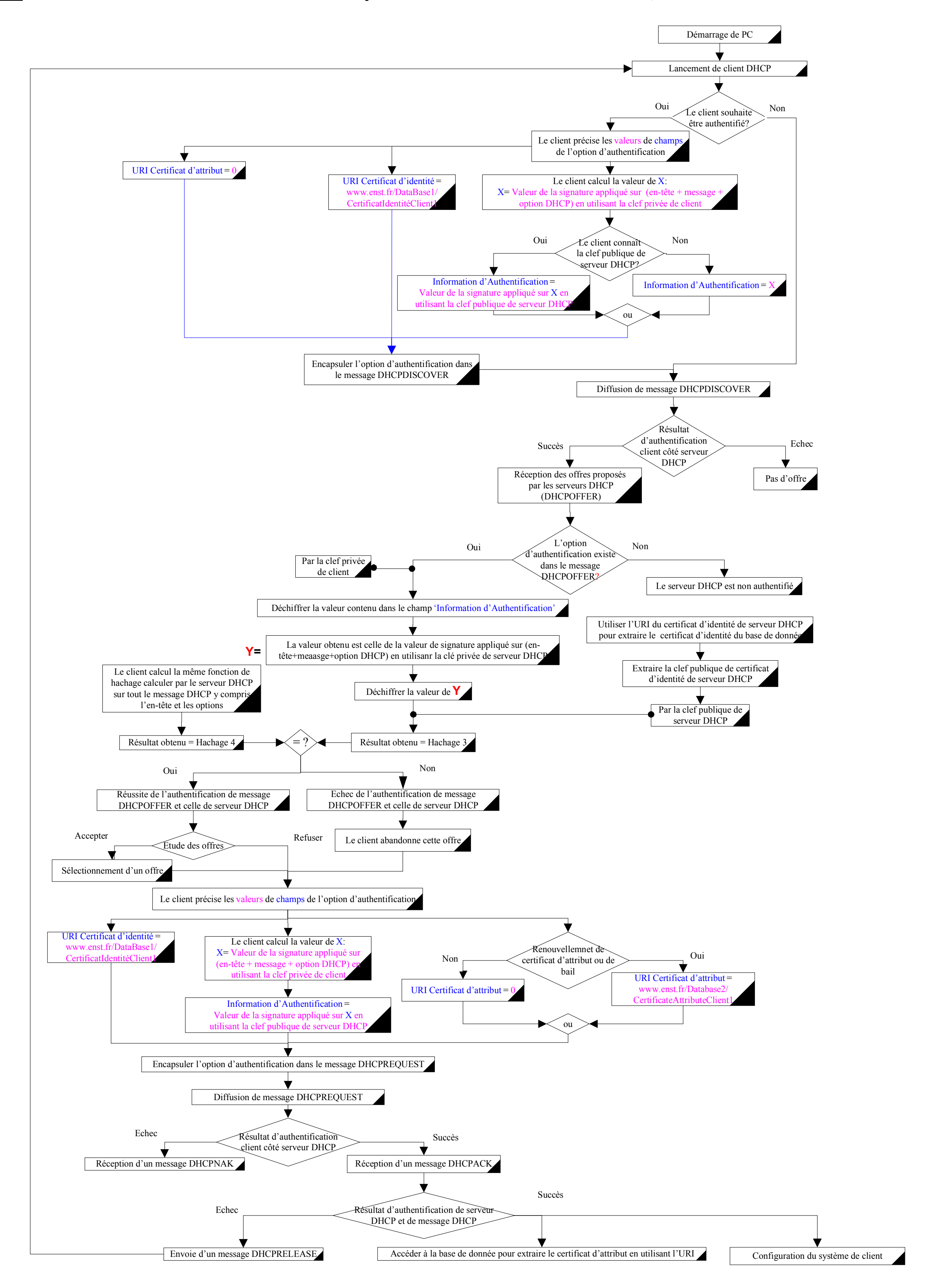


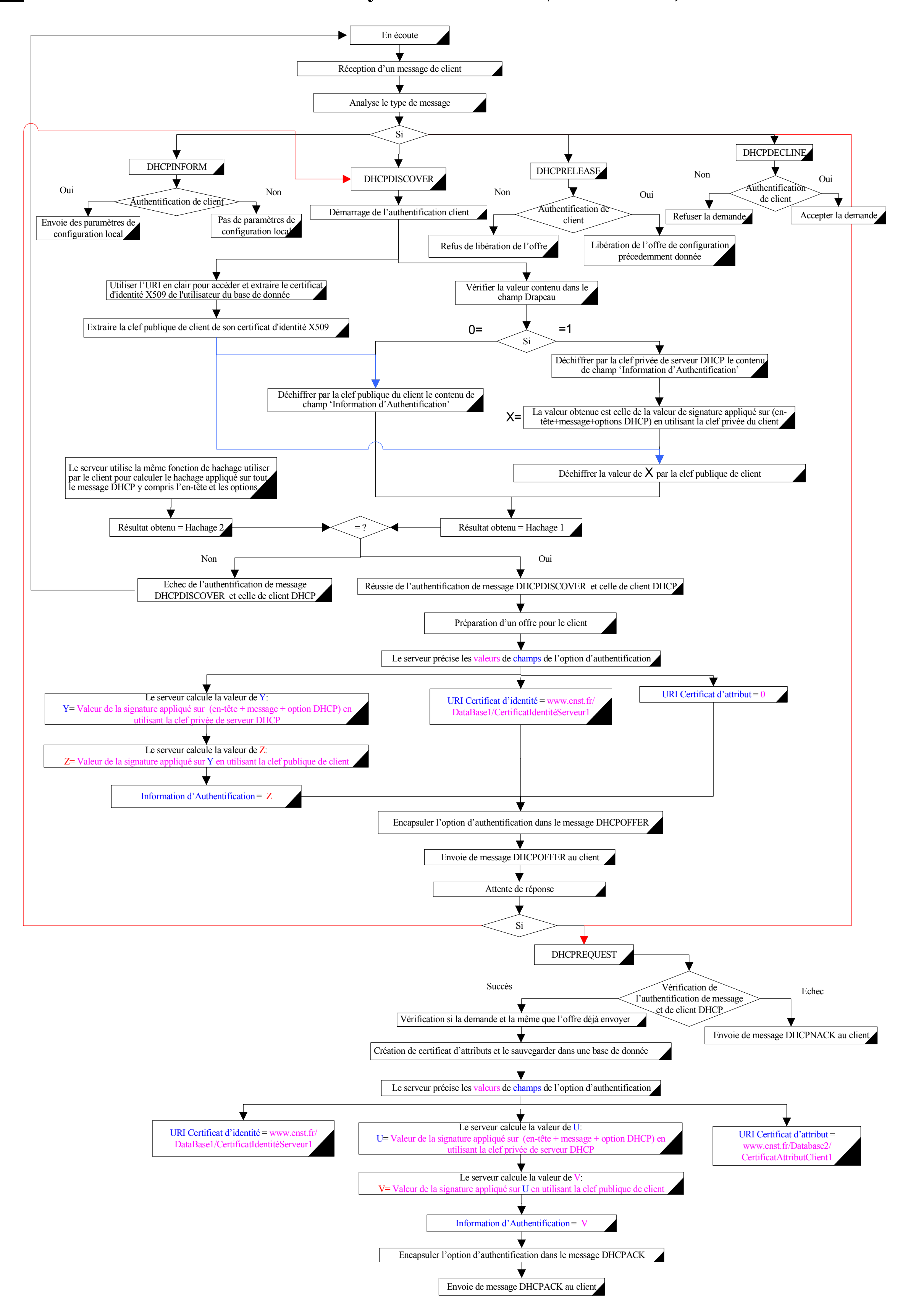
Figure 8-6 Modèle centré sur l'utilisateur

Annexe C

C.1 Vue d'ensemble du scénario de système *E-DHCP (côté Client)*



C.2 Vue d'ensemble du scénario de système *E-DHCP (côté Serveur)*



Annexe D

D ISC *DHCP* : étude du code source des logiciels client et serveur

D.1 Concernant le client

Les attributions d'adresses IP sont stockées dans un fichier de « *leases* ». Ce fichier n'est pas créé par défaut, il est donc nécessaire de le rajouter pour permettre le fonctionnement du serveur, par exemple en le créant vide :

```
touch ./work.linux-2.2/dhclient.leases
```

Le client demande un fichier de configuration, un fichier de *leases*, et un *script* de configuration correspondant à l'OS utilisé. Pour le lancer depuis le répertoire où ont été décompressées les sources, on peut taper :

```
./work.linux-2.2/client/dhclient -cf client/dhclient.conf -lf ./work.linux-2.2/dhclient.leases -sf client/scripts/linux
```

Le client n'affiche rien par défaut une fois son initialisation terminée. Les sorties standard et d'erreur sont en effet redirigées vers */dev/null*. Ceci peut être évité en excluant de la compilation une partie du code dans le fichier *client/dhclient.c* :

```
#ifdef PAS_DEFINI
        /* Close standard I/O descriptors. */
          close(0);
          close(1);
          close(2);
          /* Reopen them on /dev/null. */
          i = open ("/dev/null", O_RDWR);
          if (i == 0)
                i = open ("/dev/null", O_RDWR);
          if (i == 1) {
                i = open ("/dev/null", O_RDWR);
                log_perror = 0; /* No sense logging to /dev/null. */
          } else if (i != -1)
                close (i);
  #endif
```

Il s'agit certes d'une manoeuvre peu élégante, mais nous n'avons pas réussi à utiliser les options documentées pour que le client ne passe pas en *démon* et continue à afficher à l'écran.

D.2 Concernant le serveur

Comme le client, le serveur réclame un fichier de *leases* :

```
touch ./work.linux-2.2/dhcpd.leases
```

Pour permettre le debuggage du serveur, il est utile de le lancer au premier plan plutôt qu'en tant que *démon*. Cela se fait avec les options *-d -f* en ligne de commande. Il faut de plus un fichier de configuration et le fichier de leases. On pourra donc par exemple lancer la commande :

```
./work.linux-2.2/server/dhcpd -f -d -cf server/dhcpd.conf -lf work.linux-2.2/dhcpd.leases
```

Annexe E

E Environnement de développement

Cette partie justifie les choix techniques aussi bien au niveau matériel qu'au niveau logiciel. C'est également dans cette partie que l'on décrit précisément les interfaces des API externes utilisées pour le *E-IGP*.

E.1 Développement

L'objectif du *E-IGP* est de réaliser une application Web multi-plateforme, et il va de soi que pour rendre l'application portable l'utilisation de Java s'impose. De même que les technologies *Servlet* et *JSP* seront utilisées pour l'écriture des pages *HTML*.

Les échanges de requêtes et de certificats d'attributs utiliseront le format *XML*. Ceci procure un certain nombre d'avantage :

- Le format d'échange est en mode texte. Les requêtes ou les certificats d'attributs peuvent donc être consultés avec n'importe quel éditeur de texte.
- Le format *XML* est un format standard reconnu par le *W3C*.
- Les échanges se font par le protocole *SMTP* ce qui permet de traverser sans difficultés les barrières de sécurité existantes.

Les *API* utilisées sont les suivantes :

- *JDBC*
- *JavaMail*
- *IBM XML DSig*

Le choix de *JDBC* s'est fait pour les raisons suivantes :

- La technologie *JDBC* (*Java DataBase Connectivity*) est une *API* fournie avec *Java* (depuis sa version 1.1) permettant de se connecter à des bases de données, c'est-à-dire que *JDBC* constitue un ensemble de classes permettant de développer des applications capables de se connecter à des serveurs de bases de données (*SGBD*).
- L'*API JDBC* a été développée de telle façon à permettre à un programme de se connecter à n'importe quelle base de données en utilisant la même syntaxe, c'est-à-dire que l'*API JDBC* est indépendante du *SGBD*.
- De plus, *JDBC* bénéficie des avantages de *Java*, dont la portabilité du code, ce qui lui vaut en plus d'être indépendant de la base de données d'être indépendant de la plate-forme sur laquelle il s'exécute.

D'autre part, l'application a besoin d'envoyer des messages à des utilisateurs. Pour cela, il faut envoyer des messages à un serveur *SMTP*. L'*API JavaMail* est directement accessible sur le site *SUN*, au même titre que le *JDK*. Il remplit parfaitement l'objectif demandé, est certainement connu de la communauté des développeurs *Java*. Pour ces différentes raisons, nous l'avons choisi.

Le choix de *XML Digital Signature* (ou *XML DSig*) s'est fait pour les raisons suivantes :

- L'objectif principal de l'*E-IGP* est de démontrer l'intérêt d'utiliser les standards *XML* pour représenter les certificats d'attributs. Il semble donc naturel de s'appuyer sur les travaux de normalisation *XML* du *W3C* et de l'*IETF* pour gérer également la signature numérique de ces certificats et en particulier d'utiliser *XML Digital Signature*.

- Le standard *XML Digital Signature* est suffisamment stable maintenant, et bénéficie d'implémentations de démonstration de la part de plusieurs éditeurs (*Baltimore*, *IBM*, …) qui nous permet de rapidement mettre en œuvre cette solution.
- Nous avons fait le choix d'*IBM* en raison de la forte implication de cette société dans le standard *XML DSig* et de la disponibilité d'une plateforme de démonstration en langage *Java* (*IBM XML Security Suite*) complète et facilement réutilisable pour notre projet.

E.1.A JDBC

L'utilisation de *JDBC* se fera grâce aux interfaces et classes suivantes :

Pour manipuler les connexions vers la base de données : *java.sql.Connection.*

Pour charger le pilote de la base de données : *java.sql.DriverManager.*

Pour exécuter des requêtes SQL vers la base de données : *java.sql.Statement.*

Pour récupérer les résultats de l'exécution des requêtes SQL : *java.sql.ResultSet.*

Pour préparer une requête SQL avec pré-compilation : *java.sql.PreparedStatement.*

Pour récupérer une exception : *java.sql.SQLException.*

E.1.B JavaMail

L'utilisation de l'*API JavaMail* se fera par le biais des interfaces et classes suivantes :

1- *javax.mail.Session.*
2- *javax.mail.Message*
3- *javax.mail.internet.MimeMessage.*
4- *javax.mail.internet.InternetAddress.*
5- *javax.mail.internet.MimeBodyPart*.
6- *javax.mail.internet.MimeMultiPart*.
7- *javax.mail.Transport*.

E.1.C XML DSig

L'utilisation de l'*API XML Dsig* d'*IBM* se fera par le biais des interfaces et classes suivantes :

- Pour vérifier et signer les certificats d'attributs *XML*, on utilisera le package : *com.ibm.xml.dsig*
- Pour accéder aux primitive de sécurité *Java* et aux clefs. Les clefs sont centralisées dans le fichier *.keystore*. Sa manipulation est possible grâce à l'outil *keytool* fourni avec le *JDK* de *SUN : java.security*
- Pour accéder aux fichiers *XML :*
  1- *java.io.FileOutputStream;*
  2- *java.io.IOException;*
  3- *java.io.OutputStream;*
  4- *java.io.FileInputStream;*

E.2 Environnement

L'application web GCAX devra être multiplate-forme. Cependant, la phase de développement ainsi que la phase de pré-intégration, s'effectueront sur une plate-forme *Windows 2000*. Les tests d'intégration seront réalisées sur les plates-formes *Windows* et *Linux*.

GCAX est une application web destinée à s'intégrer dans l'infrastructure de la E-IGP, l'application nécessite donc un minimum en matière de sécurité c'est à dire le chiffrement des échanges et une identification individuelle des acteurs.

SSL permet, selon les conditions dans lesquel il est utilisé, d'assurer la confidentialité de la communication et de prévenir l'usurpation d'identité :

1- Chiffrement : seul le serveur dispose d'un certificat, celui-ci a été émis par une autorité de certification inconnue du client. La communication entre le serveur et le client peut être chiffrée. Il n'y a cependant pas d'authentification possible des parties en présence.

2- Chiffrement et authentification du serveur : seul le serveur dispose d'un certificat, et celui-ci a été émis par une autorité reconnue comme autorité de confiance par le client; le «*distinguished name*» du certificat du serveur contient un champ «*CommonName* » contenant le nom du serveur contacté. La communication est alors chiffrée et le client authentifie le serveur avec lequel il communique.

3- Chiffrement et authentification mutuelle : le serveur et le client disposent l'un et l'autre d'un certificat *X509* ; ces certificats ont été émis par des autorités reconnues par le client et le serveur. La communication est chiffrée et une authentification mutuelle du serveur et du client est effective dans la couche *SSL*.

La dernière solution a été retenue et sera mise en place grâce à l'utilisation du protocole *HTTPS* s'appuyant sur *OpenSSL*.

Le moteur de *servlet Apache Tomcat* est choisi pour les modules *GCAX* et *V&CAA* car il s'agit de l'implémentation de référence des technologies *Servlets* et *JSP*.

Le choix du *SGBD* s'est tourné vers la solution *MySQL*, car il s'agit d'un *SGBD* rapide, efficace et simple à mettre en œuvre. Ce choix n'est pas fixe dans le sens où il sera facile d'importer/exporter les données vers un autre *SGBD*.

Liste des acronymes

Liste des acronymes utilisés dans ce rapport. A chacun d'eux est associé l'explication en français ou en anglais.

AA	Attribute Authority	Autorité d'Attribut.
ACRL	Attribute Certificate Revocation List	Liste de révocation de certificat d'attributs.
API	Application Programming Interface	Interface de programmation applicative.
ARP	Address Resolution Protocol	Protocole de résolution d'adresse IPv4 en une adresse de niveau liaison (typiquement Ethernet).
ASN.1	Abstract Syntax Notation One	Notation de syntaxe abstraite numéro 1.
CA	Certification Authority	Autorité de certification.
CPS	Certification Practices Statement	Pratique de déclaration de certificat.
CRL	Certificates Revocation List	Liste de certificats révoqués.
DHCP	Dynamic Host Configuration Protocol	
DN	Distinguished Name	Nom distinctif de la norme X.500.
DNS	Domain Name System	
DTD	Document Type Definition	Définition du type de document.
FTP	File Transfer Protocol	
GCAX		Génération de Certificats d'Attribut au format XML
HTTP	Hypertext Transfer Protocol	
ICMPv6	Internet Control Message Protocol version 6	Protocole de contrôle d'IP.
ICP	Public Key Infrastructure	Infrastructure à Clefs Publiques.
IETF	The Internet Engineering Task Force	Groupe de normalisation de l'Internet.
IGC	Public Key Infrastructure	Infrastructures de Gestion des Clefs.
IGP	Privileges Management Infrastructure	Infrastructures de Gestion des Privilèges.
ILR		Ingénierie du Logiciel et des Réseaux.
IP	Internet Protocol	Protocole de niveau réseau (couche 3 du modèle OSI) utilisé dans l'Internet.
ISAKMP	Internet Security Association and key Management Protocol	Protocole de gestion des clefs et des associations de sécurité pour Internet.
ISO	International Organization for Standardization	Organisation de standardisation internationale.
ITU-T	International Telecommunication Union - Telecommunication	Union internationale pour les télécommunications.
LDAP	Lightweight Directory Access Protocol	Protocole léger d'accès à un répertoire.
LRA	Local Registration Authority	Autorité d'Enregistrement Locale.
MAC	Message Authentication Code	Code d'authentification de message.
NA	Neighbor Advertisement	Annonce de voisin dans le cadre du protocole ND. C'est l'équivalent de la réponse ARP. Un message NA vient en réponse à une requête NS.
ND	Neighbor Disvovery	Découverte des voisins. Protocole de la couche réseau, associé à IPv6. Il remplace et couvre le protocole ARP d'IPv4.
NS	Neighbor Sollicitation	Sollicitation de voisin dans le cadre du protocole ND. Permet à un équipement de résoudre l'adresse IPv6 en adresse liaison (équivalent d'une requête ARP).
OCSP	Online Certificate Status Protocol	
PKCS	Public Key Crypto Standards	Standards de crypto à clef publique.

PKI	Public Key Infrastructure	Infrastructure à Clef Publique.
PKIX	Public Key Infrastructure X.509	Infrastructure à Clef Publique basé sur le certificat X.509.
PMI	Privileges Management Infrastructure	Infrastructure de gestion de privilèges.
RA	Registration Authority	Autorité d'Enregistrement.
RFC	Request for Comment	Demande de Commentaire.
RNRT		Réseau National de Recherche en Télécommunications.
SA	Security Association	Association de sécurité.
SAA	Stateless Address Autoconfiguration	Autoconfiguration sans état.
SAD	Security Association Database	Base de données des associations de sécurité.
SPD	Security Policy Database	Base de données de politique de sécurité.
SPI	Security Parameter Index	Index des paramètres de sécurité.
SPKI	Simple Public Key Infrastructure	Infrastructure à Clef Publique Simple.
SSL	Secure Socket Layer	Couche socket sécurisée.
UTC	Universal Coordinated Time	Coordonnées horaires universelles définissant le temps selon les standards mondiaux (World Time Standard).
V&CAA		Vérification et de Contrôle d'Accès aux Applications
VPN	Virtual Private Network	Réseau Privé virtuel.
W3C	World Wide Web Consortium	Consortium du WWW.
X.500		Suite de normes ITU décrivant un système d'annuaire.
X.509		Norme décrivant le cadre de confiance applicable aux normes X.500.
XML	Extensible Markup Language	Langage de description de données défini par le W3C.